Van informatiemodel naar informatiesysteem

Leereenheid 1

Van informatiemodel naar informatiesysteem

INTRODUCTIE

Zolang er software is ontwikkeld, bestaan er softwaremodellen. Vaak hebben deze de vorm van plaatjes die fungeren als contract voor te schrijven programmacode. Als dit codeerwerk goed wordt gedaan, heeft de code precies de eigenschappen die in het model zijn gespecificeerd, plus extra (technische) eigenschappen waarover het model helemaal geen uitspraak doet, omdat ervan is geabstraheerd. Na het schrijven van de code blijft het model beschikbaar als documentatie. Maar wanneer model en code los van elkaar kunnen worden gewijzigd, is er geen garantie dat ze onderling consistent blijven. Het gevolg is dat het model niet meer als contract fungeert voor de geschreven code.

Dit is een van de problemen waarvoor *model-driven development* (MDD) een oplossing biedt. MDD is een softwareontwikkeltechnologie waarbij een model strak gekoppeld blijft aan het te bouwen softwaresysteem. Vanuit een MDD-model kan in elk stadium een werkend systeem worden gegenereerd. Zo kan het model snel en frequent worden gevalideerd bij gebruikers en andere belanghebbenden.

Deze cursus richt zich op MDD voor gegevensintensieve toepassingen (informatiesystemen). Het voorbeeld in deze leereenheid is gebaseerd op een model voor het informatiesysteem van een cd-verzamelaar. In de bijlagen wordt dit model tot leven gewekt met behulp van de MDD-tool Cathedron in de vorm van een werkende toepassing. Hierbij wordt uit hetzelfde model naar keuze een grafische Windows-interface of een webinterface gegenereerd.
In latere leereenheden wordt het model stap voor stap uitgebreid en omgebouwd, uiteindelijk tot het model voor een mediawinkel, met naast cd's ook films, boeken en games.

LEERDOELEN

Na het bestuderen van deze leereenheid wordt verwacht dat u

- een globaal idee hebt van een MDD-modelarchitectuur, in termen van informatiemodel, interfacespecificatie en bedrijfsregels
- inzicht hebt in de volgende elementen van een informatiemodel: klassen, attributen, attribuuttypen, associaties en multipliciteiten
- het verschil kent tussen een informatiemodel en een informatiediagram
- ervaring hebt opgedaan met het genereren van een informatiesysteem (database en applicatie) uit een informatiemodel
- inzicht hebt in de relatie tussen enerzijds elementen en structuur van een informatiemodel en anderzijds elementen en structuur van een hieruit gegenereerde (default) applicatie.

De studielast bedraagt circa 7 uur (3 uur voor de theorie en 4 uur voor de practica van de bijlagen).

LEERKERN

1 Wat is model-driven development?

Model-driven development MDD

Model-driven development (MDD) is een technologie om software te ontwikkelen. Bijzonder aan MDD is dat het te ontwikkelen softwaresysteem automatisch wordt gegenereerd uit een model.

1.1 MDD VOOR GEGEVENSINTENSIEVE TOEPASSINGEN

MDD is van origine een ontwikkeltechnologie voor gegevensintensieve toepassingen. In de praktijk zijn dat vrijwel altijd informatiesystemen op basis van een relationele database. De Engelse term *database driven application* geeft goed weer waar het om gaat: de database staat centraal en de applicatiefunctionaliteit wordt voor een groot deel door de database geïmpliceerd. Kernfuncties zijn de zogenaamde *CRUD*-operaties:

Database driven application

CRUD

– *Create*: aanmaken van nieuw object (in relationele terminologie: 'insert').
– *Retrieve*: ophalen van gegevens (in relationele terminologie: 'select').
– *Update*: wijzigen van een object (relationeel: een rij).
– *Delete*: verwijderen van een object (relationeel: een rij).

Ook zoekoperaties behoren tot de (voorspelbare) kernfunctionaliteit. Daarnaast kan zo'n applicatie ook niet-standaardfunctionaliteit bevatten zoals specifieke rapportages of het bewaken van allerlei regels.

Inmiddels wordt MDD ook veelvuldig toegepast bij het ontwikkelen van objectgeoriënteerde systemen (OO). In deze cursus richten we ons echter geheel op gegevensintensieve systemen op basis van een relationele database.

Andere termen
In plaats van MDD worden ook wel de termen MDE (model-driven engineering) en MDSD (model-driven software development) gebruikt. Een oudere benaming is MAD: model-based application development.

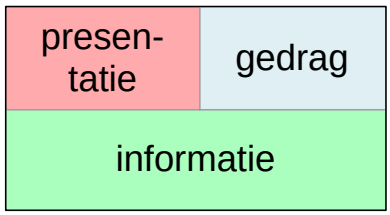
1.2 SYSTEEMGENERATIE

Omdat bij MDD een systeem automatisch wordt gegenereerd uit een model, is een MDD-model meer dan documentatie. Elke verandering in het model leidt automatisch tot een verandering in het systeem, zodanig dat model en systeem in de pas blijven lopen (zie figuur 1.1).

FIGUUR 1.1 Uit model wordt doelsysteem gegenereerd

1.3 INFORMATIE, PRESENTATIE EN GEDRAG

Aan informatiesystemen en ook aan modellen waaruit de systemen worden gegenereerd, zijn drie aspecten te onderscheiden: informatie, presentatie en gedrag. Zie figuur 1.2.



FIGUUR 1.2 Drie aspecten van een informatiesysteem

Informatie

Informatie is de basis van elk informatiesysteem. Informatie heeft verschillende gezichten. De eindgebruiker van een informatiesysteem ziet informatie via schermformulieren of papieren rapportages. Als de eindgebruiker een ander computersysteem is, 'ziet' die gebruiker de informatie misschien in de vorm van XML- of webserviceberichten. Een databasebeheerder ziet informatie als gegevens die zijn opgeslagen in een database en is niet zozeer geïnteresseeerd in de gegevens zelf als wel in de databasestructuur en de snelheid van verwerking. Een ontwikkelaar ziet het nog weer anders: die is niet in informatie geïnteresseerd, maar des te meer in de structuur ervan, los van databasekenmerken.

Presentatie

Informatie krijgt pas nut wanneer deze zichtbaar wordt gemaakt voor gebruikers. Hiermee komen we aan het tweede aspect van systeem en model: *presentatie*. We denken allereerst aan een grafische gebruikersinterface (GUI), met menu's en schermformulieren. Figuur 1.3 toont een stukje van een klantformulier, onderdeel van zo'n GUI. We merken hierbij alvast op dat een GUI een actieve interface is en daarom ook gedrag omvat.

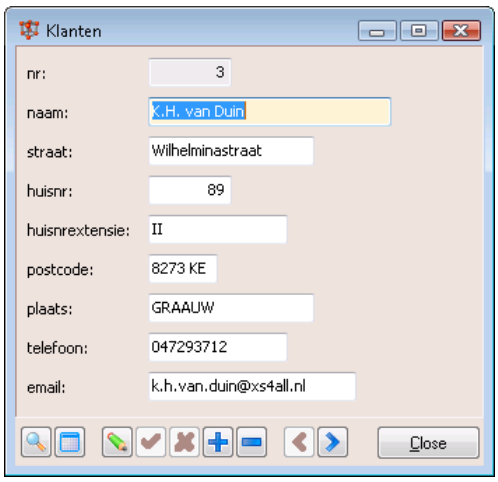
Presentatie

GUI

GUI = graphic user interface

Naast GUI's zijn er ook andere manieren van presenteren. Zo zijn er spraakinterfaces voor blinden, papieren rapportages voor managers en XML- of webservice-interfaces voor presentatie van informatie aan andere computersystemen.

We zullen ons in deze cursus voornamelijk richten op GUI's. Toch is de ruimere context van belang, want de ontwikkelfilosofie gaat uit van de gedachte dat specificaties zoveel mogelijk onafhankelijk van specifieke technologie moeten plaatsvinden. Veranderen van presentatieplatform kan dan met relatief weinig moeite plaatsvinden. We noemen hier in het bijzonder de overgang van een Windows-GUI naar een webinterface. In het practicumopdrachten zullen we dit illustreren door vanuit één specificatie twee soorten gebruikersinterfaces te genereren.



FIGUUR 1.3 GUI: presentatie en gedrag

Gedrag

Gedrag

Het derde aspect, *gedrag*, heeft betrekking op informatie en presentatie vanuit een tijdsperspectief. Allereerst zijn dat veranderingen in de tijd binnen de informatieverzameling. Denk aan het toevoegen of verwijderen van een klant of het wijzigen van klantgegevens in een database. We kunnen dit zien als pure informatiegebeurtenissen, geheel los van presentatie. Maar de GUI kan daar een belangrijke rol bij spelen als de plek waar gedrag wordt geïnitieerd. Bijvoorbeeld het klikken op een deleteknop (gebeurtenis in de GUI) leidt tot het verwijderen van een klantrecord (gebeurtenis in de informatieverzameling). Of het klikken op een knop 'Kwartaaloverzicht' leidt tot het afdrukken van een kwartaaloverzicht.
Ook veranderingen in presentatie vallen onder de noemer gedrag, bijvoorbeeld het rood worden van een rekeningveld zodra het saldo negatief wordt.

1.4 BEDRIJFSREGELS

Het informatiemodel legt een informatiestructuur vast en die structuur bevat impliciet allerlei regels. Bijvoorbeeld de regel dat een klantadres verplicht is. Wordt in het gegenereerde systeem gepoogd een klant in te voeren zonder adres, dan zal een foutmelding volgen en wordt invoer van de klant geweigerd. Dit is een vorm van gedrag. Dit gedrag zorgt ervoor dat de regel wordt gehandhaafd.

Bedrijfsregel
Business rule

Een informatiesysteem moet in het algemeen nog een flink aantal aanvullende regels bevatten, die niet al in het informatiemodel vastliggen. Deze regels worden in eerste instantie veelal in natuurlijke taal geformuleerd en later omgezet naar een formele specificatie, met het informatiemodel als context. We spreken van *bedrijfsregels* (*business rules*). Er zijn verschillende typen bedrijfsregels, waaronder afleidingsregels en constraints.

Afleidingsregel

Afleidingsregels

Een *afleidingsregel* legt vast dat een waarde uit andere waarden moet worden afgeleid en hoe dat moet gebeuren. Een voorbeeld is een btw-

bedrag, dat (automatisch) wordt berekend uit een brutoprijs en een btw-percentage, als het product daarvan.

Constraints

Constraint

Een *constraint* is een beperkingsregel, een regel die zegt dat iets niet mag. Een voorbeeld, uit de bankwereld, is de regel dat men op een spaarrekening niet rood mag staan. Zo'n constaint is zinloos, wanneer niet ook wordt vastgelegd hoe de regel wordt gehandhaafd. Hiertoe moet worden vastgelegd wat er moet gebeuren bij een overtreding of een dreigende overtreding van de regel.

Handhaving van constraints

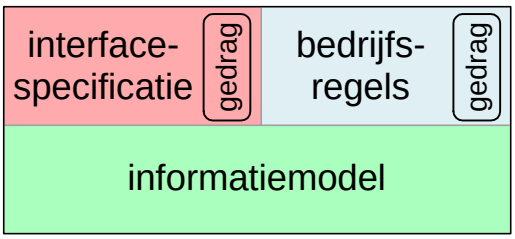
In het algemeen kan het handhaven van constraints 'hard' of 'zacht' gebeuren.
Een harde constraint wordt hard afgedwongen. Als 'niet roodstaan' een harde constraint is, moet elke transactie die tot roodstand zou leiden, worden voorkomen. Het meest plausibele gedrag is in dit geval: weigeren van zo'n transactie, onder afgifte van een melding.
Een zachte constraint wordt niet door het systeem afgedwongen, maar kan wel leiden tot een geautomatiseerde waarschuwing en eventueel tot menselijke actie. Een voorbeeld vinden we bij een kwartaalkrediet, met als constraint dat men eens in de drie maanden een positief saldo moet hebben. Dit kan natuurlijk niet door een informatiesysteem worden afgedwongen. Een klant die zich niet aan de regel houdt, veroorzaakt wel een 'ongewenste toestand', waar wat aan moet gebeuren. Wát er moet gebeuren en in welke volgorde, geautomatiseerd dan wel via menselijke actie, behoort tot het onderwerp *workflow*. We komen daar in leereenheid 9 op terug. De constraints in deze cursus zijn vrijwel allemaal harde constraints. De consequentie is dat we een voorgestelde constraint op hardheid moeten toetsen. Een veelgemaakte fout is dat een constraint die eigenlijk een workflowregel is, hard wordt geïmplementeerd. Het systeem is dan te streng en de gebruiker heeft daar last van.

Workflow

1.5 MDD-MODELARCHITECTUUR

MDD-model-architectuur

De drie aspecten informatie, presentatie en gedrag, vormen – samen met bedrijfsregels – de basis voor een MDD-*modelarchitectuur*: de opbouw van een MDD-model uit een *informatiemodel*, een *interfacespecificatie* en *bedrijfsregels* (zie figuur 1.4).



FIGUUR 1.4 MDD-modelarchitectuur

Informatiemodel

Het *informatiemodel* beschrijft de structuur van de informatie die in het systeem zal mogen (en kunnen) worden opgeslagen. Het bestaat onafhankelijk van de andere modelcomponenten en vormt de basis van het MDD-model.

Interfacespecificatie

De *interfacespecificatie* beschrijft de structuur en het gedrag van de gebruikersinterface. Dit gebeurt op een zeker abstractieniveau en is idealiter onafhankelijk van het gebruikte platform.
De interfacespecificatie verwijst naar het informatiemodel en kan niet op zichzelf bestaan.

Bedrijfsregel

Bedrijfsregels zijn aanvullende regels die nog niet zijn beschreven via het informatiemodel. Ze worden vaak in eerste instantie gespecificeerd in natuurlijke taal (zoals Nederlands) en later vertaald naar een formele specificatie in een programmeertaal. Behalve de regel zelf moet ook het gedrag worden beschreven waarmee de regel wordt gehandhaafd.

Bedrijfslogica
Business logic

De programmacode voor het handhaven van bedrijfsregels en voor niet-standaardgedrag van interfacecomponenten wordt soms *bedrijfslogica* genoemd. Vaker nog wordt de Engelse term *business logic* gebruikt. 'Logica' in deze context moet worden onderscheiden van 'logica' als de fundamentele leer van redeneren en waarheid.

Figuur 1.4 is het 'speelveld', in zijn meest grove vorm, van de hele cursus. Alle theorie behelst verfijningen of afgeleiden ervan. In leereenheid 2 gaan we dieper in op de MDD-modelarchitectuur.

2 Het informatiemodel

In paragraaf 1.5 is een MDD-softwaremodel beschreven in termen van een informatiemodel, een interfacespecificatie en bedrijfsregels. We zullen nu zien wat we ons hier bij moeten voorstellen, te beginnen met het informatiemodel. Hiertoe introduceren we eerst een 'relevante wereld' voor onze voorbeelden.

2.1 RELEVANTE WERELD

Muziekcollectie

We zullen gebruikmaken van een voorbeeld: een model genaamd Muziekcollectie1. Dit wordt opgesteld ten behoeve van een eenvoudig informatiesysteem voor de persoonlijke cd-verzameling van een muziekliefhebber, zie figuur 1.5. We beginnen eenvoudig en maken het gaandeweg complexer.

FIGUUR 1.5 Relevante wereld

Relevante wereld

De muziekcollectie vormt de *relevante wereld*. Wanneer we deze term gebruiken, bedoelen we niet de fysieke wereld van cd's, hoesjes en dergelijke, maar een geabstraheerde versie ervan. De abstractie houdt in dat we alleen kijken naar informatie over die wereld, voor zover die relevant is vanuit de optiek van de muziekliefhebber. En dat zal vaak nog lastig blijken, want wat is 'relevant'? De muziekliefhebber zal daar vooraf meestal maar een vaag idee van hebben, de informatiebehoefte zal pas tijdens het ontwikkelproces uitkristalliseren.

Usability

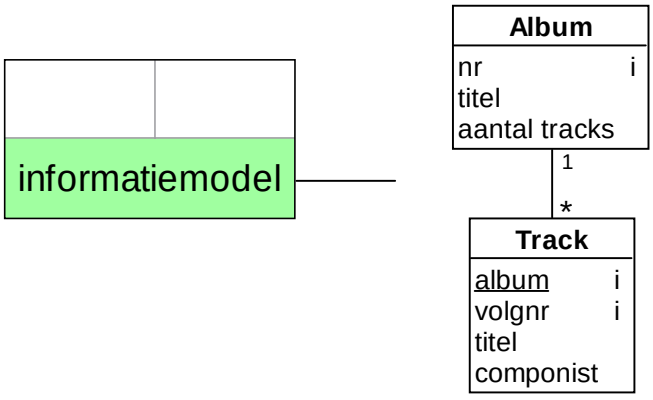
Er is nog iets in het spel: het compleet uitmodelleren van de informatiebehoefte kan op gespannen voet komen te staan met de *usability* (het gebruiksgemak) van de applicatie. Te ver uitmodelleren van details van de relevante wereld (bijvoorbeeld modelleren van alle informatie op de hoesjes) kan leiden tot een onnodig ingewikkelde applicatie. Zo kunnen we nu al zien aankomen dat de ontwikkelaar iemand is die constant dingen tegen elkaar moet afwegen. 'Het' model bestaat dan ook niet.

2.2 EEN INFORMATIEDIAGRAM

Informatiediagram

Het informatiemodel Muziekcollectie1 kunnen we niet in één keer volledig weergeven. We kunnen wel de essentie weergeven in een plaatje, een *informatiediagram* geheten. Later zullen we zien wat er nog ontbreekt om een informatiesysteem te kunnen genereren.

Figuur 1.6 geeft een eenvoudig informatiediagram bij het informatiemodel van Muziekcollectie1. Het beschrijft een informatiestructuur die betrekking heeft op albums en tracks. Het diagram is 'geschreven' in de grafische taal UML (unified modeling language).



FIGUUR 1.6 Informatiediagram bij informatiemodel Muziekcollectie1

We zullen nu de afzonderlijke elementen van het informatiediagram van figuur 1.6 bespreken.

2.3 KLASSEN EN OBJECTEN

Klasse
Object
Instantie

Het informatiediagram van figuur 1.6 geeft twee *klassen* weer, Album en Track. De klassen staan voor 'soorten van dingen': albums en tracks. Elke klasse kan worden gezien als een verzameling van *objecten*. Deze objecten

staan voor ‘dingen’: concrete albums respectievelijk tracks. Een object heet een *instantie* (‘voorkomen’) van een klasse.

Klassen en concepten
De correspondentie tussen de klassen in een model en de concepten die we in de relevante wereld onderscheiden en waar we in natuurlijke taal over praten, is allerminst eenduidig. Soms worden concepten als het ware ‘ontdekt’ en van een klasse-equivalent voorzien. Maar het komt ook vaak voor dat concepten niet tot een klasse leiden of dat verschillende concepten opgaan in één klasse.
Het is een kwestie van ervaring en van training in het doordenken van de consequenties van een keuze.

Andere termen
In teksten over informatiemodellering wordt vaak de term entiteit gebruikt, voor wat wij – UML-conform – een klasse noemen. Men komt ook de term entiteittype tegen; met ‘entiteit’ wordt dan een instantie (object) aangeduid.

2.4 ATTRIBUTEN

Attribuut

Attributen zijn eigenschappen van de objecten die tot de klasse behoren. Klassen waarmee informatie wordt gemodelleerd, en dat zijn alle klassen in deze cursus, hebben vrijwel altijd één of meer attributen.

De attributen van Album en Track
De klasse Album heeft drie attributen:
– nr, een uniek nummer (natuurlijk getal) waarmee de muziekliefhebber zelf de albums via plakkertjes heeft geïdentificeerd
– titel, voor de albumtitel
– aantal tracks, voor het aantal tracks van een album.

De klasse Track heeft vier attributen:
– album, met als waarde het albumobject waarop de track staat
– volgnr, ter onderscheiding van de track binnen het album
– titel, voor de tracktitel
– componist, voor componistinformatie.

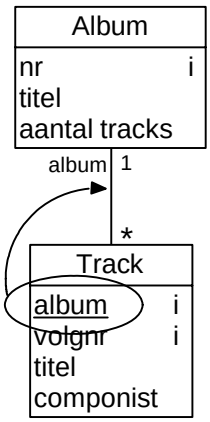
Een volledige en ondubbelzinnige attribuutaanduiding is van de vorm *klassenaam.attribuutnaam*. Bijvoorbeeld: Album.titel en Track.titel.

Namen mogen spaties of bijzonders teken bevatten.

Merk op dat de attribuutnaam ‘aantal tracks’ een spatie bevat. Zowel klassenamen als attribuutnamen mogen spaties bevatten en daarnaast bijzondere tekens, zoals é.

Associatieattribuut

Associatieattribuut
De onderstreping van het attribuut Track.album geeft aan dat dit een *associatieattribuut* is, dat is een attribuut waarvan de waarde een object is. De mogelijke waarden van Track.album zijn objecten van de klasse Album. Van elk object is de attribuutwaarde het album waarop de track staat. Zodoende hoort bij elk Track-object één Album-object. Grafisch wordt dit nog eens benadrukt door het lijntje van Track naar Album. Zie figuur 1.7.



FIGUUR 1.7 Associatieattribuut album

In figuur 1.7 is de naam van het associatieattribuut herhaald naast het corresponderende lijntje. In het algemeen zullen we zo'n herhaling achterwege laten wanneer voor de menselijke lezer duidelijk is welke klasse bij het associatieattribuut hoort. Later zullen we voorbeelden zien waarbij aan de naam van een associatieattribuut niet zo gemakkelijk is af te lezen welke klasse en welk lijntje erbij horen. Attribuutvermelding bij het lijntje kan dan verhelderend zijn. Bij invoer van een model in een MDD-tool moet de klasse bij een associatieattribuut altijd expliciet worden opgegeven. Voor de menselijke lezer echter houden we de diagrammen liefst zo kaal en overzichtelijk mogelijk.

Een Album-object kan niet worden verwijderd, zolang er een Track-object bestaat waarvan het associatieattribuut album dat object als waarde heeft. Eerst moeten dan de corresponderende Track-objecten worden verwijderd.

Cascading delete

Echter, als de MDD-tool dit ondersteunt, kan een *cascading delete* worden gespecificeerd voor het associatieattribuut, leidend tot een cascading delete voor de corresponderende verwijssleutel (foreign key) in de onderliggende relationele database. In dat geval leidt een poging tot verwijdering van een Album-object tot een poging tot verwijdering van bijbehorende Track-objecten. Als dat lukt wordt het Album-object met tracks en al verwijderd.

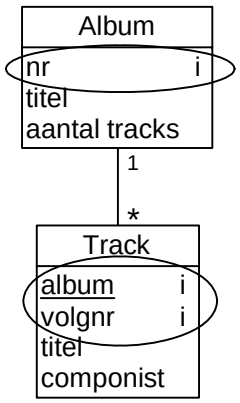
2.5 IDENTIFICERENDE ATTRIBUTEN EN ATTRIBUUTCOMBINATIES

Identificerend attribuut

Met de 'i' achter Album.nr wordt aangegeven dat dit nummer een uniek nummer is, waardoor een album wordt geïdentificeerd en onderscheiden van andere albums. Album.nr heet een *identificerend attribuut*.

Identificerende attribuut-combinatie

Per album is het volgnummer van een track uniek, zodat de combinatie van de attributen Track.album en Track.volgnr identificerend is voor een track. Dit wordt aangegeven met twee i's. De combinatie Track.album, Track.volgnr heet een *identificerende attribuutcombinatie*. Zie figuur 1.8.



FIGUUR 1.8 Identificerend attribuut en identificerende attribuutcombinatie

Identificatieregel

Het identificerend zijn van een attribuut of attribuutcombinatie is in feite een regel, een *identificatieregel*. Deze omvat twee deelregels:

Verplichte-waarderegel

– het attribuut of de attribuutcombinatie moet een waarde hebben (*verplichte-waarderegel*)

Uniciteitsregel

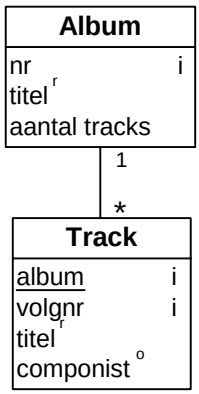
– alle attribuutwaarden of waardencombinaties moeten verschillend zijn (*uniciteitsregel*).

Per klasse zullen we altijd precies één identificatieregel geven, die – zoals we nog zullen zien – medebepalend is voor de automatische transformatie van informatiemodel naar database. Het ligt voor de hand te veronderstellen dat een identificerend attribuut of een identificerende attribuutcombinatie wordt getransformeerd naar een primaire sleutel van een databasetabel. In het algemeen is dit echter geen juiste veronderstelling. Het kan bijvoorbeeld zijn dat voor de databases tabellen met aparte, kunstmatige primaire sleutels worden gegenereerd. Of dat dat alleen gebeurt voor klassen met een samengestelde identificatieregel.

2.6 OPTIONELE EN VERPLICHTE ATTRIBUTEN

Een *verplicht attribuut* is een attribuut waarvan de waarde door de gebruiker verplicht moet worden ingevuld (verplichte-waarderegel). In een informatiediagram kunnen we dit expliciet aangeven (het hoeft niet!) met ^{r} (van 'required'). In figuur 1.9 gebeurt dat voor de attributen Album.titel en Track.titel.

Tegenover een verplicht attribuut staat een *optioneel attribuut*, daarvan mag de waarde niet-ingevuld blijven (in databasetermen: *null*). Wanneer we in een informatiediagram optionaliteit expliciet willen aangeven, doen we dat met ^{o}. In figuur 1.9 is Track.componist optioneel. Het diagram zegt dat er (behoudens andere regels) tracks zonder componistinformatie mogen voorkomen.



FIGUUR 1.9 Optioneel, verplicht of 'buiten beschouwing'

Let op: geen r of °, dan geen conclusie trekken

We vinden het belangrijk de diagrammen zo eenvoudig mogelijk te houden, daarom zullen we in de cursustekst r en ° alleen vermelden wanneer we erover hebben nagedacht en vermelding toegevoegde communicatiewaarde heeft. Staat er niets bij een attribuut, dan mogen we geen conclusie trekken over verplicht zijn of optionaliteit. Het enige wat we mogen concluderen, is dat de opsteller van het diagram het niet nodig vond hierover iets op te nemen. Misschien wist die nog niet of het een verplicht attribuut moest worden. Of misschien wist die het wel, maar wilde die het in dit diagram niet vermelden, om het eenvoudig te houden. In figuur 1.9 is 'Album.aantal tracks' zo'n attribuut.

identificerend attribuut verplicht tenzij anders vermeld

Op het voorgaande maken we één uitzondering, maar ook dat is weer voor de eenvoud: een identificerend attribuut (i) is verplicht, tenzij optionaliteit expliciet wordt aangegeven (°). Dat laatste is alleen mogelijk voor een attribuut dat deel uitmaakt van een identificerende attribuutcombinatie. In figuur 1.9 zijn dus zowel het identificerende attribuut van Album verplicht als de beide attributen van de identificerende attribuutcombinatie van Track. Later zullen we enkele voorbeelden tegenkomen van klassen met een identificatie door twee attributen waarvan er één optioneel is.

UML-profile
Onze UML-notatie is een 'profile' met afwijkingen van standaard UML. Zo duidt onderstreping in ons profile een associatieattribuut aan. Standaard UML kent geen associatieattributen. Een tweede, belangrijke afwijking van standaard UML is het gebruik van identificerende attributen.

Identificatieregels en objectidentificatie in OO-systemen
In objectgeoriënteerde programmeeromgevingen kennen we het begrip *objectidentificatie*. Dit houdt in dat elk object, ongeacht zijn attribuutwaarden een eigen objectidentiteit heeft. Twee verschillendeobjecten, met elk hun eigen objectidentiteit, kunnen dus dezelfde attribuutwaarden hebben. Bij objectidentificatie door middel van attribuut-identificatieregels is dat onmogelijk: verschillende objecten verschillen altijd in hun identificerende attribuutwaarde(n). Dit heeft alles te maken met het feit dat we bij MDD modelleren voor gegevensintensieve, databasegeoriënteerde systemen.

OPGAVE 1.1
Zullen in een informatiesysteem dat conform het model van figuur 1.9 wordt gebouwd, albumtitels op een eenduidige, gestandaardiseerde manier worden opgeslagen? En de namen van componisten?

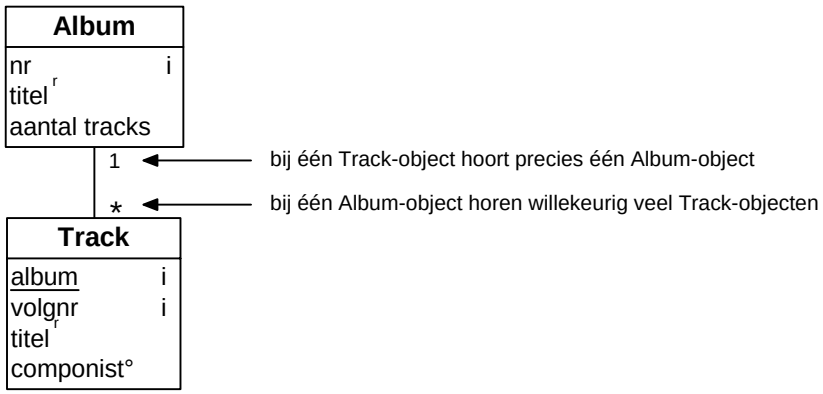
2.7 ASSOCIATIES EN MULTIPLICITEITEN

Associatie

Het associatieattribuut Track.album legt een correspondentie vast tussen de klasse Track en de klasse Album: bij een Track-object hoort één Album-object en bij een Album-object kunnen willekeurig veel (nul of meer) Track-objecten horen. Deze correspondentie draagt de naam *associatie*. In het informatiediagram wordt de associatie uitgedrukt door het verbindingslijntje tussen de twee klassen.

Multipliciteit
Cardinaliteit

De aantalaanduidingen 'precies één' of 'nul of meer' heten *multipliciteiten* of *cardinaliteiten* en worden grafisch aangegeven met de 1 respectievelijk * bij het verbindingslijntje (zie figuur 1.10).



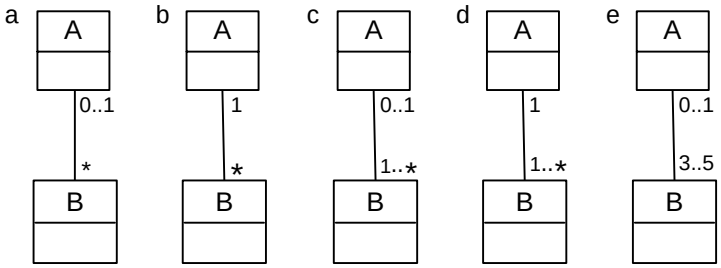
FIGUUR 1.10 Multipliciteiten

Let op de leesvolgorde: bij een Track-object hoort precies één (1) Album-object. Lees de 1 aan de 'overkant' van het lijntje. Evenzo: bij een Album-object horen nul of meer, dat wil zeggen willekeurig veel (*) Track-objecten.
Andere mogelijke multipliciteiten zijn: 0..1 (nul of één), 1..* (één of meer) of bijvoorbeeld 3..6 (minimaal 3, maximaal 6).

Minimum multipliciteit
Maximum multipliciteit

De 1 is eigenlijk een verkorte notatie voor 1..1 en * een verkorte notatie voor 0..*. Vóór de punten staat steeds de *minimum multipliciteit* en erachter de *maximum multipliciteit*, waarbij * staat voor 'willekeurig veel'.

Figuur 1.11 geeft een aantal voorbeelden van associaties tussen willekeurige klassen A en B, met hun multipliciteiten.



FIGUUR 1.11 Voorbeelden van 1-op-n-associaties

1-op-n-associatie
n-op-1-associatie

In plaats van de * (de UML-notatie) treft men vaak ook n aan. Alle associaties van de figuren a t/m e noemen we *1-op-n-associaties* (vanuit A gezien) of *n-op-1-associaties* (vanuit B gezien).

Ouderklasse
Kindklasse

De klasse aan de 1-kant wordt *ouderklasse* of *parent class* genoemd en de klasse aan de n-kant *kindklasse* of *child class*. We zien: bij een ouderobject horen 'n' kindobjecten, bij een kindobject horen nul of één ouderobjecten.

Belangrijke tekenconventie

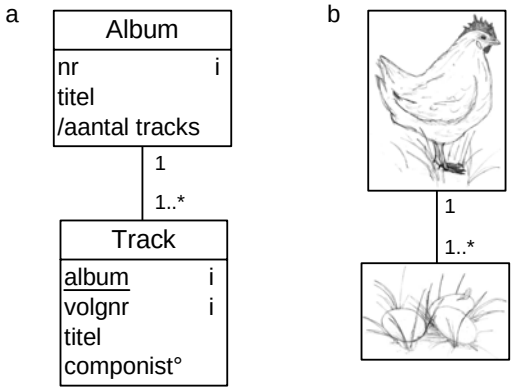
De ouderklasse tekenen we vrijwel altijd hoger dan de corresponderende kindklassen. Dit is een belangrijke tekenconventie. Later zullen we zien dat deze conventie ons toestaat de multipliciteiten in veel gevallen weg te laten.

Elke associatie tussen klassen A en B kunnen we opvatten als een verzameling objectparen (a, b), waarbij a een A-object is en b een B-object.
In leereenheid 3 introduceren we ook n-op-m-associaties.

OPGAVE 1.2
Stel, er zijn 100 B-objecten. Geef voor elk van de figuren 1.12 a t/m e minima en maxima voor:
a het mogelijke aantal A-objecten
b het mogelijke aantal A-objecten dat is geassocieerd met een B-object
c het mogelijke aantal elementen (objectparen) van de associatie.

2.8 EEN KIP-EI-PROBLEEM

Figuur 1.11d stelt ons voor een probleem: bij elk A-object hoort minimaal één B-object en bij elk B-object hoort precies één A-object. Hoe krijgen we dat voor elkaar uitgaande van een beginsituatie zonder A- en B-objecten? Figuur 1.12a maakt de situatie wat concreter voor albums en tracks. De vraag is hoe we een eerste album met track geregistreerd krijgen, zonder een van de multipliciteitsregels te overtreden. Beginnen we met het album, dan overtreden we de regel '1 of meer tracks', beginnen we met de track dan overtreden we de regel '1 album'. Het lijkt een kip-ei-probleem: 'elke kip komt uit een ei, maar om een ei te krijgen heb je een kip nodig'. Zie figuur 1.12b.



FIGUUR 1.12 Een kip-ei-probleem

Aangezien we een album en een track niet tegelijkertijd kunnen invoeren, is hier alleen maar uit te komen wanneer één van de regels tijdelijk niet hoeft te gelden. Dat is mogelijk, maar in de regel alleen wanneer we de 1-op-1..*-regel niet via het informatiemodel afdwingen, maar op een andere manier, via de applicatie. De details zijn tool- en databaseafhankelijk. In blok 3 leert u hier meer over.

'Wereld' en 'relevante wereld'

Kip-ei-problemen zijn allerminst triviaal, ook in omgevingen met geavanceerde transactiemechanismen (zoals alle tegenwoordige relationele databases). Dat is al reden genoeg om in het informatiemodel kip-ei-multipliciteiten te vermijden. Maar los van een technisch probleem moeten we ons ook de vraag stellen of een 1-op-1..* associatie echt is wat we willen.

Niet 'de wereld' modelleren maar de informatiebehoefte

Het is verleidelijk om te poneren dat in werkelijkheid elk album minstens één track heeft en dat *dus* de multipliciteit aan de n-kant 1..* moet zijn. Echter, bij informatiemodelleren gaat het er niet om hoe 'de wereld' in elkaar zit, maar wat de informatiebehoefte over die wereld is. Dat is precies wat met 'relevante wereld' wordt uitgedrukt.
De vraag die we ons moeten stellen is dus niet of elk album tracks heeft en ook niet of trackinformatie wenselijk is maar of we het onmogelijk willen maken albuminformatie op te slaan zonder trackinformatie. We moeten ons realiseren dat een verbod erg ver gaat en dat we daar misschien spijt van krijgen. (De trackinformatie is wellicht niet altijd direct beschikbaar of misschien wil de muziekliefhebber eerst alle albumnummers, -namen en -componisten kunnen invoeren en pas later de tracks.)

Als een verbod inderdaad te ver gaat, is ons probleem opgelost: we modelleren niet 1-op-1..* maar 1-op-*. Desgewenst kunnen we in de applicatie nog een waarschuwing inbouwen, wanneer een album wordt ingevoerd zonder tracks.
Als we voor een verbod kiezen, moeten we in het informatiemodel echter eveneens kiezen voor 1-op-*. In dat geval is echter helder dat we dat alleen doen om een technisch probleem te omzeilen, het 'kip-ei'-probleem. Dit betekent niet dat we afzien van de beoogde multipliciteitsregel. Het betekent alleen dat de 1-op-1..*-regel niet via het informatiemodel kan worden afgedwongen.

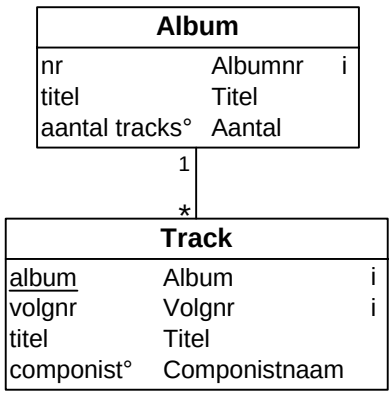
2.9 ATTRIBUUTTYPEN

Attribuuttype

Elk attribuut in het model van Muziekcollectie1 heeft een verzameling toegestane waarden. Deze verzameling heet het *attribuuttype* van het attribuut. Zonder attribuuttypen is het model onvolledig en is het onmogelijk er een database uit te genereren. Databasetabellen hebben immers kolommen met een bepaald datatype (zoals Integer of Varchar) en die datatypen moeten afgeleid kunnen worden uit het informatiemodel.

In onze informatiediagrammen zullen we het attribuuttype meestal weglaten, om ruimte te besparen en omdat het attribuuttype in eerste instantie minder belangrijk is. Vooral bij grotere diagrammen bevordert dit het overzicht. Dit neemt niet weg dat attribuuttypen van groot belang zijn en zorgvuldig gekozen moeten worden.

Voorbeeld: Muziekcollectie1
Als voorbeeld bekijken we opnieuw het informatiediagram van Muziekcollectie1, maar nu uitgebreid met attribuuttypen (zie figuur 1.13).



Attribuuttype

naam	*datatype*
Albumnr	Integer
Titel	Varchar (100)
Aantal	Integer
Album	(klasse)
Volgnr	Integer
Componistnaam	Varchar (50)

FIGUUR 1.13 Informatiediagram Muziekcollectie1, met attribuuttypen

De figuur toont elk attribuut met de naam van het bijbehorende attribuuttype. De namen van attribuuttypen zullen we steeds met een hoofdletter schrijven.
We zullen nu alle attributen met hun attribuuttypen kort nalopen.

Om te beginnen heeft attribuut Album.nr een attribuuttype met de naam Albumnr. In de attribuuttypetabel is dit gedefinieerd als een verzameling van gehele getallen (datatype Integer).

Eén attribuuttype voor meerdere attributen

Het attribuut Album.titel heeft een attribuuttype genaamd Titel, met als datatype tekst van variabele lengte (Varchar) tot een maximum van 100. Dit attribuuttype wordt ook gebruikt voor het attribuut Track.titel. Dat ligt wel voor de hand: een tracktitel is net zo'n soort waarde als een albumtitel.

Het attribuut 'Album.aantal tracks' heeft een attribuuttype genaamd Aantal. Evenals Albumnr zijn de waarden hiervan gehele getallen. Er is

voor een apart attribuuttype gekozen, omdat een aantal tracks een ander soort getal is dan een albumnummer.

Wanneer we attribuut Track.album even overslaan, resten nog Track.volgnr en Track.componist. Track.volgnr krijgt een eigen attribuuttype Volgnr. Kennelijk wordt een tracknummer als een ander soort waarde gezien (met mogelijkerwijs andere eigenschappen) dan een albumnummer. Ook Track.componist krijgt een eigen attribuuttype: Componistnaam. Het datatype hiervan is weer Varchar, met een geschikte lengte.

Attribuuttype van een associatieattribuut
Het attribuut Track.album heeft – als associatieattribuut – een speciaal attribuuttype, namelijk de klasse Album. De waarden hiervan zijn Album-objecten. Dit wordt ook uitgedrukt door de onderstreping, in combinatie met het verbindingslijntje tussen Track en Album.

In het algemeen geldt: het attribuuttype van een associatieattribuut is een klasse. Het attribuuttype van andere attributen is een 'gewoon datatype'.

In het kader van een strakke, eenvoudige naamgeving geven we een associatieattribuut vaak dezelfde naam als de klasse waarnaar wordt verwezen, maar met een kleine letter.

Keuze van attribuuttypen

Regel voor attribuuttypen

Voor semantisch verschillende attributen definiëren we verschillende attribuuttypen. De reden hiervoor is dat we dan onderscheid kunnen maken tussen bijvoorbeeld een klantnummer en een ordernummer, door ze elk een eigen attribuuttype te geven. Semantisch hebben we het over totaal verschillende nummers maar een klantnummer kan de waarde 34 hebben en een ordernummer ook.

Voor de applicatie zijn deze verschillende attribuuttypen ook gunstig: per attribuuttype kunnen we een aantal eigenschappen (properties) instellen zoals bijvoorbeeld de defaultwaarde en de displaylengte op een formulier. Als we klantnummer en ordernummer zouden baseren op één attribuuttype Nummer, dan zou een instelling voor dit attribuuttype consequenties hebben voor beide attributen, wat in het algemeen onwenselijk is.
Voor semantisch gelijksoortige attributen kunnen we wel een gemeenschappelijk attribuuttype gebruiken, bijvoorbeeld een attribuuttype Datum voor geboortedata en overlijdensdata of een attribuuttype Geldbedrag voor verkoopprijs, inkoopprijs of salaris.

OPGAVE 1.3
We zagen dat attributen (voor zover geen associatieattributen) een datatype hebben, bijvoorbeeld Varchar(100), via hun attribuuttype. Wat is het voordeel hiervan boven het rechtstreeks toekennen van zo'n datatype aan elk attribuut?

2.10 WERKWIJZE

Het informatiemodel van figuur 1.13 bevat twee klassen met daartussen een 1-op-n-associatie. De klasse aan de 1-kant is de ouderklasse, die aan de n-kant is de kindklasse. Later zal blijken dat elk informatiemodel

geheel is opgebouwd uit dit soort bouwstenen. Wanneer we weten in welke volgorde de klassen, attributen en attribuuttypen bij een 1-op-n-associatie het beste kunnen worden ingevoerd, weten we dat ook voor grotere informatiemodellen.

We kunnen dit probleem op twee manieren benaderen: vanuit de structuur van het ontwerp en vanuit de praktijk van het ontwerpen. Deze manieren leiden tot lichtelijk verschillende werkwijzen.

Werkwijze vanuit de structuur
Stel we hebben een model met klassen, attributen en attribuuttypen op papier geschetst, zoals het diagram van figuur 1.13. We zien: een attribuut hoort tot een klasse en is van een of ander attribuuttype. Vóór we een attribuut invoeren, moeten dus eerst zijn klasse en zijn attribuuttype zijn gecreëerd.
Speciale aandacht verdienen de associatieattributen: hun attribuuttype is zelf een klasse (Track.album heeft als type Album). In het algemeen zal daarom de ouderklasse moeten worden gecreëerd vóór de kindklasse.
Van belang zijn ook de identificatieregels. Zo'n regel kan worden gecreëerd zodra het attribuut of de attribuutcombinatie waarvoor de regel moet gelden is gecreëerd.

Stappenplan creatie ouder- en kindklasse

Dit leidt tot het volgende stappenplan voor de creatie van twee klassen met daartussen een 1-op-n-associatie.

1 Creëer de attribuuttypen van de ouderklasse.
2 Creëer de ouderklasse met haar attributen en identificatieregel.
3 Creëer de attribuuttypen van de kindklasse (voor zover nog niet gebeurd).
4 Creëer de kindklasse met haar attributen (waaronder het associatieattribuut bij de ouder) en identificatieregel.

Omdat we consequent de ouderklasse boven de kindklasse tekenen, creëren we op deze manier de klassen 'van boven naar beneden'.
Deze werkwijze komt overeen met de manier waarop een eindgebruiker gegevens in de database invoert: deze kan pas een track invoeren als het bijbehorende album in de database zit.

Werkwijze vanuit de ontwerppraktijk
We geven nog een tweede stappenplan dat iets meer recht doet aan de praktijk van het ontwerpen. De attribuuttypen komen hierbij pas zo laat mogelijk in beeld, namelijk bij invoer van een getekende schets in de MDD-tool.

1 Schets en bediscussieer een klassenstructuur met de belangrijkste attributen (waaronder de associatieattributen).
2 Creëer voor elke ouder/kindcombinatie eerst de ouder en dan het kind, als volgt:
a Creëer de ouderklasse met haar attributen; kies voor elk attribuut het attribuuttype uit een keuzelijst of creëer een nieuw attribuuttype 'just in time'.
b Creëer een identificatieregel voor de ouderklasse.
c Creëer analoog de kindklasse met haar attributen (waaronder het associatieattribuut) en identificatieregel.

Aangepast stappenplan ouder- en kindklasse

Dit stappenplan maakt dus onderscheid tussen een schetsfase (op papier of whiteboard, waarbij attribuuttypen nog geen rol spelen) en een creatiefase waarbij de schets wordt ingevoerd in de MDD-tool en de attribuuttypen 'just in time' bekend moeten zijn.

3 Bedrijfsregels en interfacespecificatie

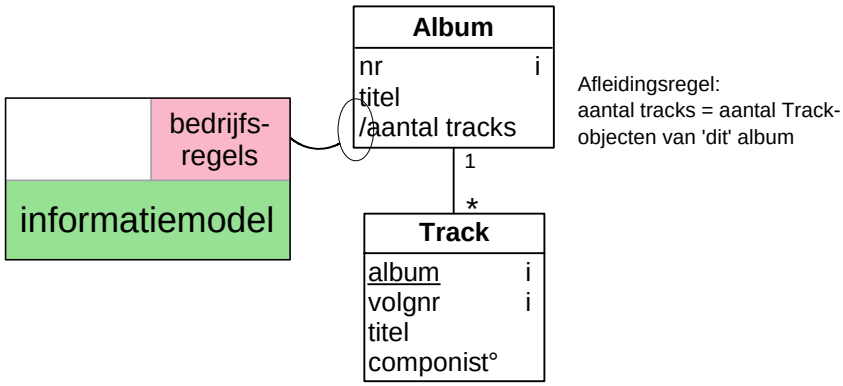
Het informatiemodel is het belangrijkste onderdeel van een MDD-model. Het is de basis ervan. We gaan nu kort in op de andere onderdelen: bedrijfsregels en de interfacespecificatie. In blok 3 van deze cursus komen deze uitgebreid aan de orde.

3.1 EEN BEDRIJFSREGEL

Bedrijfsregel

Afleidbaar attribuut

In figuur 1.14 is het informatiediagram uitgebreid met een *bedrijfsregel* in natuurlijke taal (Nederlands). De bedrijfsregel is een afleidingsregel, die zegt dat de waarde van aantal tracks *afleidbaar* is en dat de berekende waarde is wat de naam belooft: het aantal Track-objecten van het betreffende album. Een afleidbaar attribuut wordt weergegeven door een slash (/).
Bedrijfsregels vormen een aparte component van een MDD-model, naast het informatiemodel, zoals figuur 1.14 toont.



FIGUUR 1.14 Informatiediagram uitgebreid met afleidingsregel

Hoe het attribuut afgeleid moet worden, moeten we nader specificeren in de bedrijfsregelcomponent van het model. Het diagram van figuur 1.14 is hierin dus nog onvolledig. Er zijn minstens twee mogelijkheden:

– hertelling van het aantal Track-objecten na elke toevoeging van een nieuw Track-object of na verwijdering van een Track-object
– beginnen met aantal tracks = 0 en met 1 ophogen na toevoeging van een Track-object en met 1 verlagen na verwijdering van een Track-object.
Ongeacht de keuze moet natuurlijk nog voorkomen worden dat een gebruiker de waarde van aantal tracks 'zomaar' kan wijzigen.

Bedrijfsregels zullen we veelal op een informele manier al 'meenemen' bij het informatiemodel. Het bewaken ervan (de reactie van het systeem op een dreigende overtreding) is echter een verhaal op zich. We zijn daar pas aan toe in de leereenheden 9 en 10.

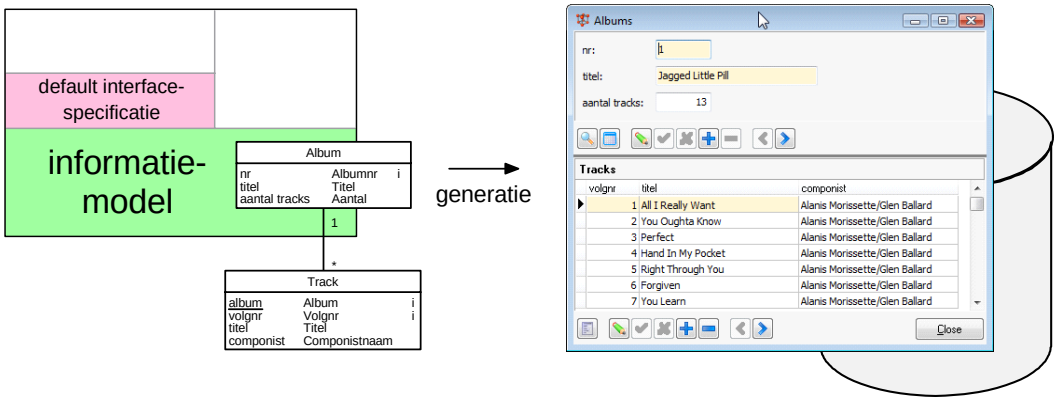
3.2 INTERFACESPECIFICATIE

Interfacespecificatie

De *interfacespecificatie* beschrijft de gebruikersinterface. Het specificeert de 'aanblik' van het systeem voor de gebruiker: menu's, formulieren, knoppen, enzovoort. Impliciet wordt daarmee ook veel gedrag gespecificeerd, want het is een 'levende' interface die uit het model wordt gegenereerd (zie figuur 1.4).

Default interfacespecificatie

Veel functionaliteit van een informatiesysteem is voorspelbaar en ligt al min of meer vast door het informatiemodel. Dat geldt bijvoorbeeld voor de CRUD-operaties en zoekfuncties (zie paragraaf 1.1). Daarom bieden MDD-tools de mogelijkheid om zonder expliciete interfacespecificatie een compleet informatiesysteem te genereren. Hiervoor is slechts een informatiemodel met klassen, attributen en attribuuttypen nodig. De MDD-tool genereert hieruit schermen op basis van een *default interfacespecificatie*, leidend tot een standaard ('default') informatiesysteem. Zie figuur 1.15.



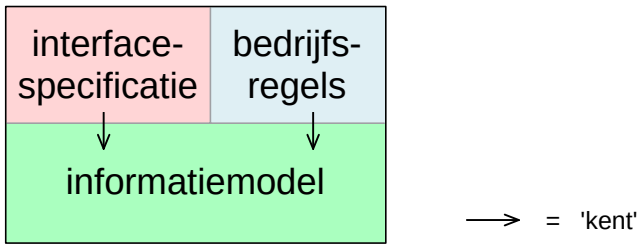
FIGUUR 1.15 Default informatiesysteem gegenereerd op basis van informatiemodel en default interfacespecificatie

In paragraaf 4 gaan we hier nader op in. De interfacespecificatie is naar believen aan te vullen met eigen specificaties, leidend tot een interface met alle gewenste formulieren, menu's, enzovoort, zoals we zullen zien in leereenheid 9.

3.3 SAMENHANG VAN DE DRIE MODELCOMPONENTEN

Figuur 1.16 geeft de onderlinge samenhang aan van de drie modelcomponenten: informatiemodel, interfacespecificatie en bedrijfsregels.

– Het informatiemodel bevat geen referenties naar interfacespecificatie of bedrijfsregels; het bestaat geheel op zichzelf.
– Interfacespecificatie en bedrijfsregels staan niet op zichzelf, ze bevatten referenties naar het informatiemodel.



FIGUUR 1.16 Samenhang: welk onderdeel kent welk ander onderdeel?

Merk op dat interfacespecificatie en bedrijfsregels onafhankelijk van elkaar zijn (details zoals schermpjes voor foutmeldingen bij overtreding van een bedrijfsregel buiten beschouwing gelaten).

3.4 STAPPENPLAN VAN EEN PROJECT

Figuur 1.16 impliceert dat het informatiemodel minstens voor een deel gereed moet zijn, voordat de corresponderende delen van de interfacespecificatie en van de set bedrijfsregels kunnen worden opgesteld. In de 'workflow' van een project, dat wil zeggen de door de ontwikkelaar te verrichten taken met hun volgorde, gaat het opstellen van het informatiemodel dus vooraf aan de andere taken. Een stappenplan, geïdealiseerd, ziet er aldus uit:

stap 1 opstellen van het informatiemodel
– aanmaken van een nieuw project
– creatie van klassen, hun attributen en daarvan de attribuuttypen
– generatie van database en (default) applicatie

stap 2 uitbouwen van de gebruikersinterface:
– specificatie gebruikers(groepen), menustructuren en bijbehorende formulieren.

stap 3 toevoegen van bedrijfsregels:
– specificatie in natuurlijke taal
– omzetting in formele taal
– implementatie (business logic)

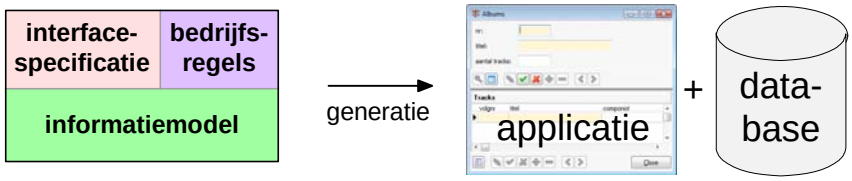
In de praktijk worden deze stappen iteratief (in een herhalingsproces) doorlopen, waarbij het model en het bijbehorende systeem incrementeel (in stapjes) groeien.

Stappenplan bij associatie

Voor twee klassen waartussen een associatie bestaat kunnen we deze workflow detailleren met een van de stappenplannen van paragraaf 2.10.

4 Genereren van database en applicatie

In figuur 1.1 werd geschetst hoe uit een MDD-model een werkend systeem ('doelsysteem') kan worden gegenereerd. Inmiddels weten we dat een MDD-model drie componenten kan omvatten : een informatiemodel, een interfacespecificatie en bedrijfsregels. Dit leidt tot een nadere invulling van de figuur 1.1, zoals gegeven in figuur 1.17.



FIGUUR 1.17 Generatie van doelsysteem uit drie-componentenmodel

De interface kan zowel een Windows-GUI als een webinterface zijn, naar keuze van de ontwikkelaar.

4.1 DEFAULT INFORMATIESYSTEEM

Wanneer de bedrijfsregelcomponent 'leeg' is en de interfacespecificatie beperkt is tot de default interfacespecificatie, is figuur 1.17 gelijkwaardig met figuur 1.15.
De default applicatie is een mooi tussenstation waarmee een eindgebruiker kan experimenteren en waarmee het informatiemodel kan worden gevalideerd.

4.2 OUDER EN KIND, MASTER EN DETAIL

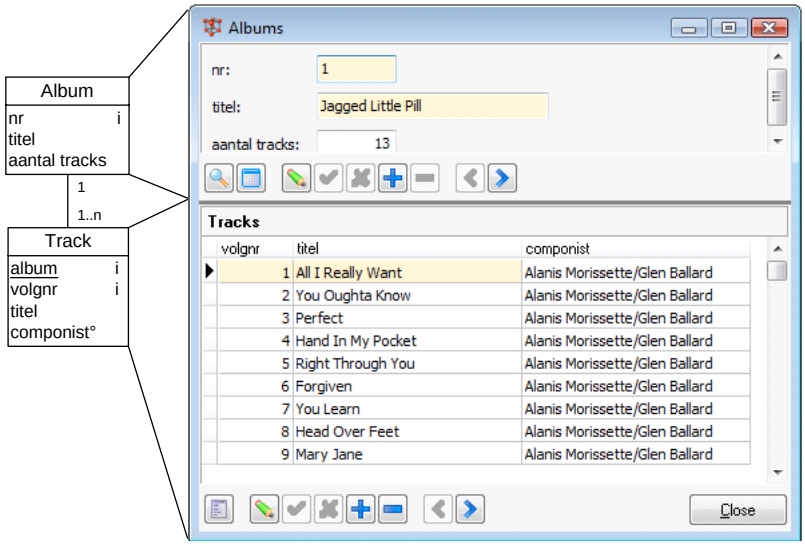
Master-detailformulier

Welke MDD-tool ook wordt gebruikt, de gegenereerde applicaties zullen altijd in essentie op hetzelfde neerkomen. Zo zal bij een klasse altijd een formulier worden gegenereerd voor het onderhoud van de betreffende gegevens. En er zal een menu worden gegenereerd om formulieren te kunnen kiezen. Soms kunnen met geringe toevoegingen aan het informatiemodel (die feitelijk op het gebied van de interfacespecificatie liggen) de mogelijkheden van de applicatie aanzienlijk worden verruimd en dat met een minimum aan moeite.

Bij een 1-op-n-associatie zal het default formulier van de ouderklasse vrijwel altijd de vorm hebben van een *master-detailformulier*, met een subformulier voor de kindklasse.

Het formulier zichtbaar in figuur 1.15 is zo'n master-detailformulier, voor de klasse Album. In figuur 1.18 is het nog wat duidelijker weergegeven, ook in correspondentie met het informatiediagram. Het formulier bestaat uit twee 'frames', één voor albums (het masterframe) en één voor de bijbehorende tracks (het detailframe). Het Track-frame is gesynchroniseerd met het Album-frame. Daarom wordt het eerste Track-attribuut (associatieattribuut album) niet getoond in het Track-frame. Het is immers al zichtbaar in het Album-frame.

In bijlage 3 van deze leereenheid (zie ook opdracht 1.2) wordt aangegeven hoe u de default-applicatie bij Muziekcollectie1 realiseert in Cathedron, de MDD-tool bij deze cursus. Daarbij worden de stappenplannen van paragraaf 2.7 gebruikt voor de invoer van klassen, attributen en attribuuttypen in de juiste volgorde.



FIGUUR 1.18 Master-detailformulier voor albums-met-tracks in default applicatie

MDD-tools

Model-driven development is een technologie om snel en flexibel software te ontwikkelen. Veel hangt af van de kracht van de gebruikte MDD-tool. Maar wat is kracht? We zouden kunnen zeggen: hoe minder de ontwikkelaar hoeft te doen om een bepaald resultaat te bereiken, des te krachtiger is de tool. Een krachtige MDD-tool werkt met modellen van een hoog abstractieniveau, met zo min mogelijk technologiespecifieke details. Alles wat specifiek is voor een technologie (platform of architectuur), wordt idealiter door de tool gegenereerd. Dit ideaal wordt echter door geen enkele MDD-tool bereikt. Daarvoor is de complexiteit van databases (Oracle, SQL Server, MySQL, …), applicaties (Java, C♯, C++, …) en de wisselwerking daartussen te groot.

Cathedron

Zie www.cathedron.com.

In deze cursus werken we met de MDD-tool *Cathedron*. Deze is ontwikkeld vanuit de praktijk én de behoeften van het onderwijs, in samenwerking met de Open Universiteit. De tool leent zich goed voor snel prototypen en is daarmee een uitstekende tool voor validatie van informatiemodellen en ondersteuning van het leren modelleren.

OPDRACHTEN

Deze leereenheid bevat drie bijlagen met praktische opdrachten:

In bijlage 1 wordt verteld hoe Cathedron te installeren en op te starten en hoe in te loggen. U wordt wegwijs gemaakt in de menustructuur voor de ontwikkelaar en de (default) menustructuur voor de eindgebruiker.

Via bijlage 2 verkent u de default applicatie Muziekcollectie1 in de rol van eindgebruiker.

Via bijlage 3 verkent u het project Muziekcollectie1 in de rol van ontwikkelaar. U begint met een blanco project, voert het informatiemodel in en genereert een database met default applicatie.

TERUGKOPPELING

Uitwerking van de opgaven

1.1 Albumtitels worden op één plaats opgeslagen, bij het unieke albumnummer. Ze liggen op die manier eenduidig vast. Voor de componistnamen ligt dit anders. Deze wijzen niet naar een klasse; ze zijn slechts een tekstwaarde bij een track. Zodra een componist vaker voorkomt, wordt deze meervoudig opgeslagen, zonder garantie op eenduidigheid. Van standaardisatie is daardoor geen sprake.

1.2 a Bij de figuren 1.12b en d is er minimaal één A-object. In dit uiterste geval zijn alle 100 B-objecten met dit ene A-object geassocieerd. Bij de figuren a, c en e kan het minimum ook 0 zijn.
In de gevallen a en b is er geen maximum voor het aantal A-objecten. In de gevallen c en d zijn er maximaal 100 A-objecten. In geval e is het maximum 33.
b Minima-maxima: 0-100, 1-100, 0-100, 1-100, 0-33.
c Minima-maxima: 0-100, 100, 0-100, 100, 0-100.

1.3 De attributen Album.titel en Track.titel illustreren het nut. Beide hebben het attribuuttype Titel met datatype Varchar(100). De ontwikkelaar heeft hiermee uitgedrukt dat het om dezelfde soort titels gaat. Blijkt op zeker moment dat Varchar(100) een te klein datatype is, dan wordt dit *op één plaats* in het model veranderd, waarna langere titels voor zowel tracks als albums beschikbaar zijn.
Als de ontwikkelaar tot de conclusie komt dat albumtitels en tracktitels verschillend behandeld moeten worden, dan moet worden gekozen voor verschillende attribuuttypen.

Bijlage 1

Cathedron – installatie en opstarten

Installatie Cathedron en MDD-voorbeelden
Via de cursussite kunt u twee installatieprogramma's downloaden en uitvoeren:
– installatieprogramma van Cathedron
– installatieprogramma van de voorbeelprojecten.

Voor beide neemt een wizard u bij de hand. Hierbij zijn enkele punten om even bij stil te staan.

Installatie Cathedron

Licentieovereen-komst

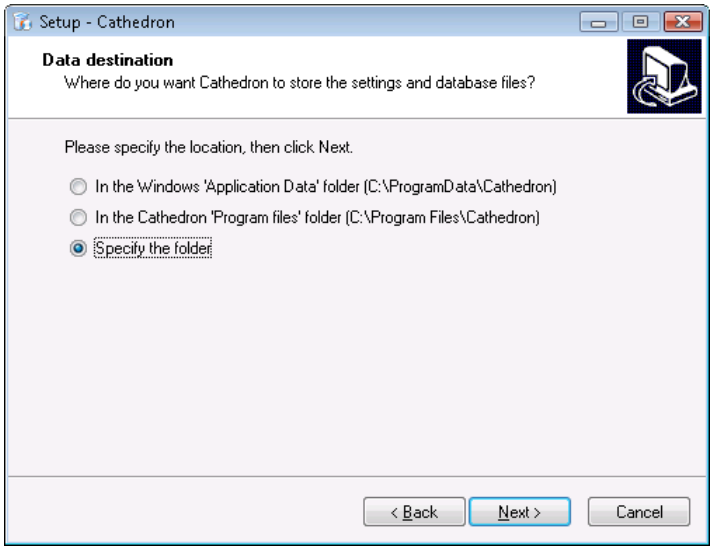
Allereerst de licentieovereenkomsten. Lees deze goed door. Het belangrijkste punt is dat het gebruik van Cathedron voor onderwijsdoeleinden vrij is, maar voor commerciële doeleinden niet.

Locatie Cathedron en projecten

Een volgend punt betreft de locatie waar Cathedron geïnstalleerd zal worden. Hiervoor kunt u de defaultkeuze volgen, de Program Files-map.

Projectmap

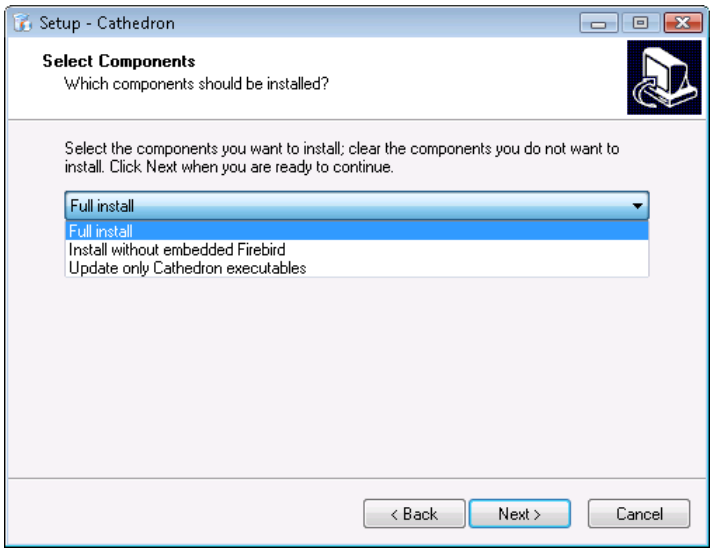
Verder is er een vraag naar de map waar de projecten komen te staan. De cursus gaat uit van installatie in C:\Cathedron, omdat dit een eenvoudig en kort pad is. Wilt u dit pad gebruiken of een ander eenvoudig pad instellen, kies dan voor de optie Specify the folder (zie figuur 1) en geef de gewenste locatie op. Kies u voor één van de standaardlocaties, zorg er dan voor dat u deze locatie later terug kunt vinden in de Verkenner.



FIGUUR 1 Locatie voor database files (projecten).

Selectie componenten

Het vierde punt om even bij stil te staan, is de selectie van componenten, zie figuur 2. Kies voor Full install, als u Cathedron voor het eerst installeert. Wanneer u echter een nieuwere versie van Cathedron over een oudere heen wilt installeren, is het zaak te kiezen voor Update only Cathedron executables. In dat geval blijven al uw Cathedron-projecten ongewijzigd.



FIGUUR 2 Te installeren onderdelen

Installatie met of zonder embedded database

Het laatste punt betreft het onderliggende databaseprogramma, Firebird. Mocht u al ervaring hebben met Firebird, dan kunt u er ook voor kiezen om Firebird apart te installeren. In dat geval kunt u tijdens de installatie van Cathedron kiezen voor Install without embedded Firebird. Zorg er in dat geval wel voor dat u in Firebird een gebruiker DEV met wachtwoord dev aanmaakt.

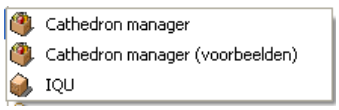
Installatie MDD-voorbeelden

Wanneer u de MDD-voorbeelden installeert, worden deze geïnstalleerd in de map C:\Cathedron\Projects\Examples, als u ten minste bij installatie van Cathedron hebt gekozen voor C:\Cathedron als projectmap.

Opstart en login

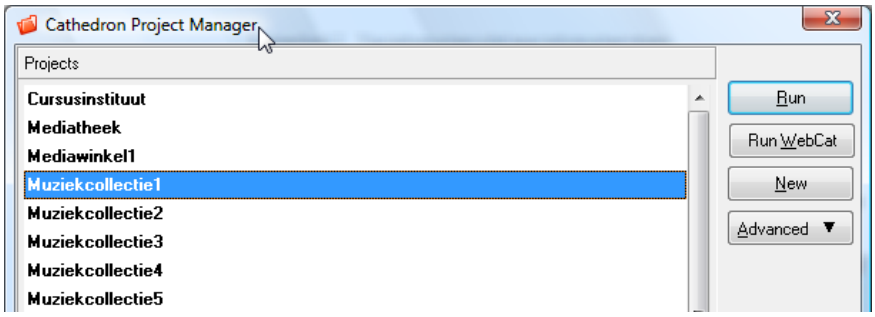
Cathedron starten we op door middel van de Cathedron (project)manager via Start | Programma's | Cathedron. Voor de voorbeelden uit de cursus gebruikt u de optie met voorbeelden, voor eigen projecten de eerste optie, zie figuur 3. De menuoptie IQU in deze figuur staat voor *Interactive Query Utility*, de SQL-editor van Cathedron.

Interactive Query Utility



FIGUUR 3 Cathedron opstartopties

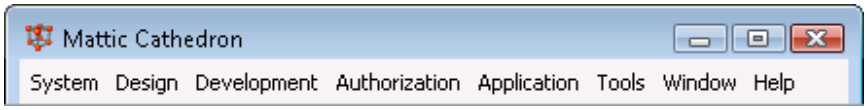
Start, door aanklikken van Cathedron manager (voorbeelden), de Cathedron Project Manager. U kunt nu een van de voorbeelden van deze cursus selecteren en op Run klikken, zie figuur 4.



FIGUUR 4 Cathedron Project Manager met Muziekcollectie1 geselecteerd

Cathedron start dan op met het gekozen project en met de gebruiker in de rol van ontwikkelaar. Cathedron toont een verder leeg venster met een menustructuur.

Applicatiemenu en ontwikkelmenu
Als u Cathedron opstart op de hiervoor beschreven manier, doet u dat in de rol van ontwikkelaar. Cathedron toont dan een menustructuur specifiek voor de rol van ontwikkelaar (zie figuur 5).



FIGUUR 5 Menustructuur voor de ontwikkelaar

Tijdens het ontwikkelen kan overgeschakeld worden naar de menustructuur zoals een eindgebruiker die ziet. Cathedron genereert default menu's: voor ieder default formulier is er een menuoptie (zie figuur 6).

FIGUUR 6 Menustructuur voor de eindgebruiker

Het overschakelen gaat via System | Application menu en terug met System | Development menu.
Een eindgebruiker start Cathedron op via een loginvenster en heeft uiteraard alleen toegang tot het applicatiemenu.
Als u Cathedron opstart met de eerste optie 'Cathedron manager' van figuur 3, moet u ook inloggen, zie figuur 7.

FIGUUR 7 Inlogvenster van Cathedron

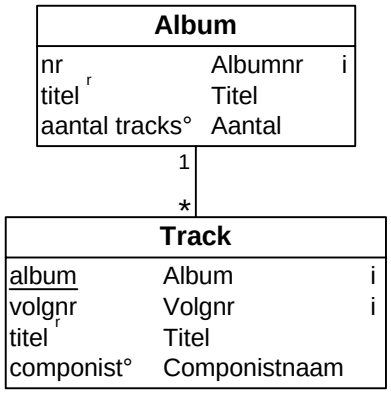
De benodigde inloggegevens voor een ontwikkelaar zijn:

Name: DEV (u mag kleine letters intypen)
Password: dev

Bijlage 2

Cathedron als applicatieomgeving (practicum)

In deze bijlage bekijken we de default applicatie bij Muziekcollectie1, gebaseerd op het informatiediagram van figuur 1. De applicatie is kant-en-klaar meegeleverd. We kruipen in deze bijlage dus in de rol van gebruiker.



Attribuuttype

naam	*datatype*
Albumnr	Integer
Titel	Varchar (100)
Aantal	Integer
Album	(klasse)
Volgnr	Integer
Componistnaam	Varchar (50)

FIGUUR 1 Informatiediagram van Muziekcollectie1

Figuur 1 combineert de figuren 1.10 en 1.14. Het is een volledig gespecificeerde versie, waarin niets in het midden is gelaten. De identificerende attributen zijn, volgens conventie (zie paragraaf 2.4), verplicht. Het attribuut 'aantal tracks' is optioneel gemaakt. De reden is dat we dit attribuut later afleidbaar zullen maken (zie paragraaf 3.1), waardoor het automatisch wordt berekend. Het moet dan in eerste instantie leeg kunnen blijven.

Open nu het project Muziekcollectie1 in de Cathedron Manager (voorbeelden).

In bijlage 3 zult u deze applicatie zelf ontwikkelen.

Menu's
Voor een eindgebruiker bestaat de interface van een Cathedron-applicatie uit een menustructuur waarmee formulieren opgeroepen kunnen worden. In eerste instantie wordt een default menu gegenereerd. Zie bijvoorbeeld figuur 6 van bijlage 1: het eindgebruikersmenu van de ontwikkelde applicatie (Muziekcollectie1).
In leereenheid 9 zullen we ook non-default menu's leren maken, één voor elke verschillende gebruiker of gebruikersgroep.

Een ander voorbeeld van een menu is figuur 5 van bijlage 1: het menu voor de ontwikkelaar. Bedenk hierbij dat de ontwikkelaar zelf ook

eindgebruiker is, namelijk van de ontwikkelapplicatie. Die ontwikkelapplicatie is zelf ook een Cathedron-applicatie, dat wil zeggen ooit ontwikkeld in Cathedron.

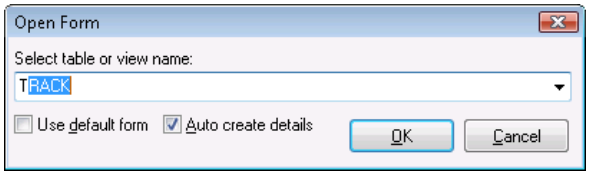
We zullen in deze bijlage werken via het ontwikkelaarsmenu van figuur 5 van bijlage 1. Desgewenst kunt u de formulieren ook bereiken via het eindgebruikersmenu van figuur 6 van bijlage 1.

'Cathedron is ontwikkeld in zichzelf'
Je kunt stellen dat 'Cathedron is ontwikkkeld in Cathedron'. Nauwkeuriger geformuleerd: Cathedron is ontwikkeld via een eenvoudiger versie van Cathedron. Die eenvoudiger versie is zelf in een nog eenvoudiger versie ontwikkeld, enzovoort. Ooit is begonnen met een oerversie, die is geprogrammeerd in Delphi (een ontwikkelomgeving die op zijn beurt is gebaseerd op Object Pascal).

Formulieren
Via formulieren worden gegevens getoond en wordt de mogelijkheid geboden – via een knoppenbalk – om gegevens te zoeken, toe te voegen, te verwijderen of te wijzigen.

De default applicatie bestaat uit een aantal formulieren, een voor iedere klasse uit het informatiemodel. Ieder formulier heeft dezelfde naam als de bijbehorende klasse. De default applicatie van Muziekcollectie1 kent dus twee formulieren: Album en Track.

Een van de manieren om een formulier te openen, is via Tools | Open Form: selecteer een formulier uit de afrollijst of type de beginletter, zie figuur 2. Alternatief: via het default eindgebruikersmenu (knop Application).



FIGUUR 2 Selectie van het Track-formulier

Merk op dat figuur 2 melding maakt van 'table' en niet van 'class'. Die 'table' is een tabel van de onderliggende relationele database die gelijknamig is met de klasse.

Sneltoetsen

Snel en efficiënt werken met de tool is bij MDD belangrijk. Niet alleen in de praktijk maar ook al binnen deze cursus. U zult merken dat alle moeite die u daarvoor doet, snel wordt terugverdiend. U kunt bijvoorbeeld overwegen voor veel voorkomende taken sneltoetsen te gebruiken. U vindt deze in de menu's.

Full table scan

Full tabel scan
In de educatieve versie van Cathedron die met deze cursus wordt meegeleverd, wordt na het openen van een formulier een *full table scan* uitgevoerd, waarbij alle records van de betreffende tabel naar de client worden gestuurd en in het formulier worden getoond. In het ontwikkelstadium en bij kleine voorbeeldpopulaties is dit prettig. Voor de productieversie van Cathedron zou

dit echter, zeker bij grote tabellen, ongewenst gedrag zijn, vanwege onnodige belasting van resources van client en server en omdat het getoonde eerste record zelden het gezochte record zal zijn en een zoekactie dus tóch nog moet volgen.

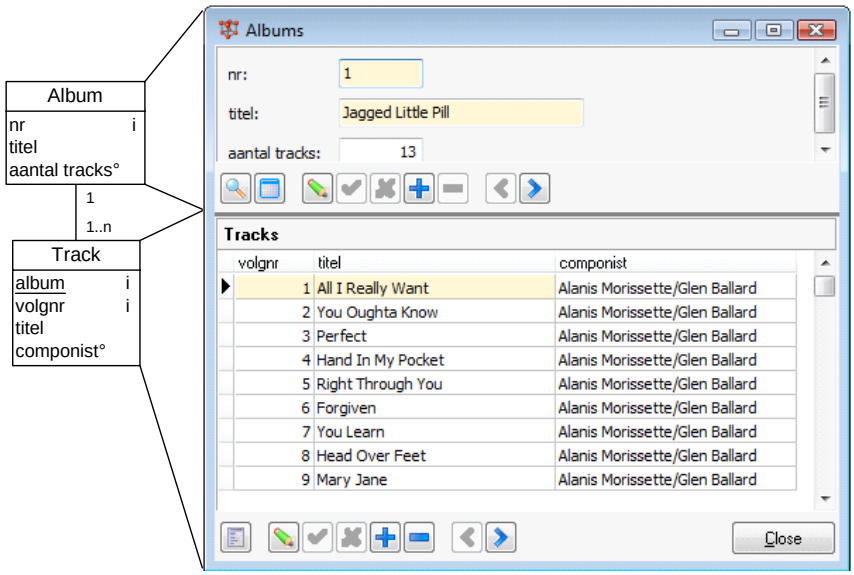
Master-detailformulier

Master-detailformulier

Er bestaat een 1-op-n-associatie tussen Album en Track. Deze zien we terug in het default formulier bij Album, een *master-detailformulier* (zie figuur 3, die identiek is met figuur 1.20). Voor elk album worden niet alleen de gegevens van dat album (het masterrecord), maar ook die van de bijbehorende tracks (de details) getoond.

Frame

Cathedron toont in twee *frames* de gegevens van Album met bijbehorende tracks.



FIGUUR 3 Master-detailformulier voor Album

De formuliertitel (Albums) is de meervoudsvorm van de klassenaam. Ieder attribuut van het informatiemodel is zichtbaar op een formulier, afgezien van attribuut album.

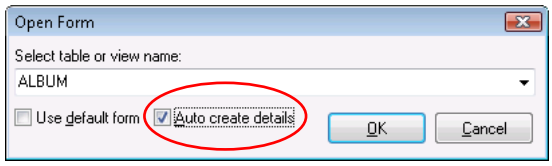
Kleuren

Toelichting op de achtergrondkleuren van de velden:
– wit: optionele kolom
– rose/oranje: verplichte kolom.
– grijs: niet editbaar (een voorbeeld volgt in figuur 11)

Opmerkingen

Het detailgedeelte heeft geen zoekknop (vergrootglas). Verderop komt het zoeken aan de orde.
Het associatieattribuut Track.album wordt niet afgebeeld in het detailgedeelte. Afbeelden zou niet zinvol zijn, omdat de waarden overeenkomen met die van het identificerende attribuut Album.nr in het mastergedeelte.

Een masterformulier openen zonder details
Soms is het handig een formulier te openen zonder de bijbehorende detailframes. Om dit te realiseren, haalt u in het Open Form-formulier het vinkje voor Auto create details (zie figuur 4) weg.



FIGUUR 4 Wel/niet automatisch genereren van detail-frames

Single record view
Multi record view

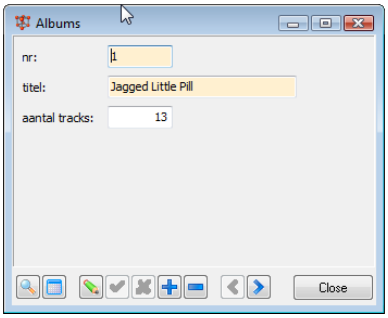
Grid

Overschakelen van view

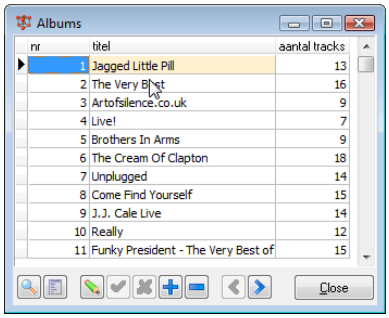
Single record view en multi record view
Standaard is het bovenste frame van een master-detailformulier in *single record view* (srv), dat wil zeggen met één record per frame. Het onderste staat in *multi record view* (mrv), dat wil zeggen met meerdere records in een tabelopmaak. Multi record view wordt ook wel *grid* genoemd. Overschakelen van de ene view naar de andere doet u door op het symbool naast het vergrootglas te klikken.

In figuur 5 ziet u twee versies van een Album-formulier, beide zonder details. Formulier a is in srv, formulier b in mrv.

a



b

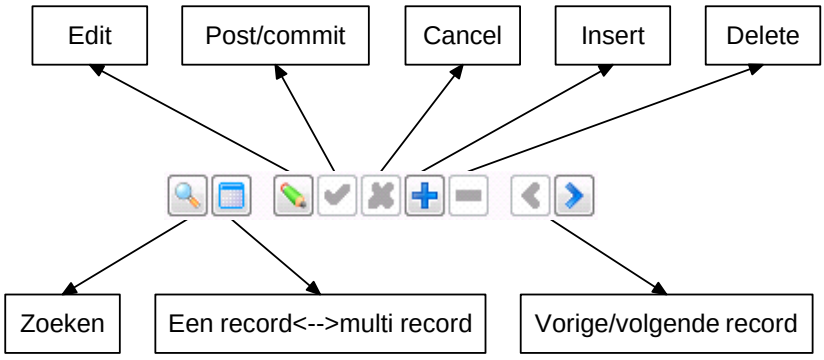


FIGUUR 5 Single record view (a) en multi record view (b) van Album

Open formulier Album. (Probeer eventueel meerdere manieren uit.) Schakel over van view in zowel het master- als het detailframe. Ga na hoe het zit met de synchronisatie. Hoe kunt u in beide views zien dat er meerdere records beschikbaar zijn? Experimenteer ook met een Album-formulier zonder details.

Aan de actieve (blauwe) navigatieknoppen in srv kunt u zien dat u met meerdere records te maken hebt. In mrv zien we dat direct in het formulier. Aan de schuifbalk aan de rechterzijde is te zien dat er nog meer records zijn dan er getoond worden.

Gegevens invoeren en bewerken
Onder in de formulieren Album en Track zien we een werkbalk met knoppen. Deze knoppen representeren een stuk default functionaliteit. Figuur 6 toont de betekenis van iedere knop. Als een knop grijs is, kan deze op dat moment niet gebruikt worden.



FIGUUR 6 De knoppen van de werkbalk

Dataset

Post

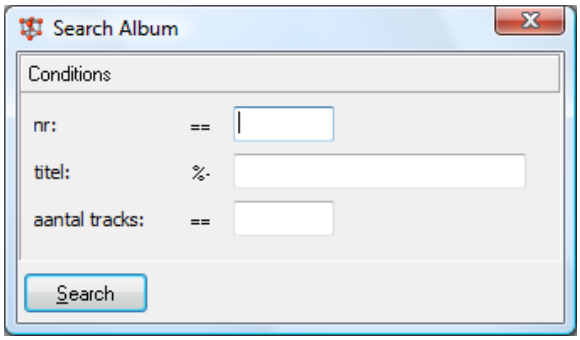
'Post' en 'posten' zijn Engelse woorden!

Om de werking van de knoppen te begrijpen, is het belangrijk te weten dat in een applicatieformulier getoonde gegevens meestal een afbeelding zijn van een kopie van gegevens in de database. Die kopie heet een *dataset*. (Datasets zijn niet specifiek voor Cathedron.)
Als u op de knop Edit klikt, gaat u in eerste instantie wijzigingen aanbrengen in deze dataset. Klikt u daarna op knop Post/Commit, dan worden de gewijzigde gegevens naar de database *gepost*. Als de *post* wordt geaccepteerd, volgt direct een commit. De database bevat hierna daadwerkelijk de gewijzigde gegevens. Bij de postoperatie worden alle geldende bedrijfsregels gecontroleerd.
De knop Insert geeft u de mogelijkheid in het formulier en in de dataset gegevens voor een nieuw record in te voeren, maar ook hier geldt dat pas na Post/Commit het nieuwe record aan de database wordt toegevoegd.
Met de knop Cancel wordt een lopende edit- of insertoperatie geannuleerd.
De knop Delete verwijdert een rij uit het formulier en de dataset, waarna deze verwijdering direct wordt gepost naar de database. Een commit volgt, tenzij er een constraint is die dit verhindert.

CRUD-operaties, post en commit
De knoppen corresponderen deels met de vier basisoperaties op een tabel, die vaak worden aangeduid met het acroniem CRUD: zie paragraaf 1.1.
Een post en een commit zijn in Cathedron (en in vele andere omgevingen) gecombineerd in één knop. Dit dwingt korte, eenvoudige databasetransacties af, wat veel fouten voorkomt en dus een groot voordeel is.

Zoeken
Direct na het openen van een formulier toont Cathedron u – in de educatieve versie van de tool – de gegevens van de bijbehorende tabel. Zoals we in paragraaf 2.1 opmerkten, is dit gedrag ongewenst in een productieomgeving; het is echter erg prettig in een ontwikkelcontext met kleine tabellen.
Via de navigatietoetsen kunt u door de records bladeren. Bij een grote tabel is deze manier van navigatie niet geschikt om een bepaald record

op te zoeken. Beter is dan te zoeken via de knop met het vergrootglas, waarbij Cathedron een zoekformulier opent (zie figuur 7).



FIGUUR 7 Zoekformulier voor tabel Album

Het zoekformulier bevat tussen kolomnamen en editvelden knopjes voor keuze van de gewenste zoekfunctionaliteit. De opschriften zijn een symbolische weergave van zoekoperatoren, sommige voor numerieke gegevens, andere voor alfanumerieke. Pas als u met de muiscursor in de buurt komt van zo'n opschrift, ziet u dat het om een knopje gaat.
Tabel 1 toont de diverse zoekopties voor de standaard datatypen die Cathedron (eigenlijk Firebird) ondersteunt.

TABEL 1 Zoekopties voor verschillende doorzoekbare datatypen

soort	*teken*	*betekenis*	
alfanumeriek	`%-`	starts with	(default)
	`=`	equal (case insensitive)	
	`==`	equal (case sensitive)	
	`<`	less	
	`>`	greater	
	`<=`	less or equal	
	`>=`	greater or equal	
	`..`	between	
	`%`	contains	
	`-%`	ends with	
	`#`	empty	
	`!`	not empty	
	`<>`	not equal	
datum, tijd en numeriek	`==`	equal	(default)
	`<`	less	
	`>`	greater	
(pop-up kalender beschikbaar voor datuminvoer)	`<=`	less or equal	
	`>=`	greater or equal	
	`..`	between	
	`#`	empty	
	`!`	not empty	
	`<>`	not equal	
memo	`%`	contains	(default)
	`=`	equal (case insensitive)	
	`-%`	starts with	
	`%-`	ends with	

Blob

Naast de getoonde datatypen kent Cathedron nog het type *blob* ('binary large object'). Blobs worden gebruikt voor bijvoorbeeld plaatjes of geluidsfragmenten. Als een attribuut op dat type gebaseerd is, ziet u dat niet terug in een zoekvenster: blobs staan niet toe dat ermee gezocht wordt. Gezien de aard van dit datatype is dat heel logisch.

Als u op het knopje van de zoekoperator klikt, krijgt u de mogelijkheid deze te wijzigen.

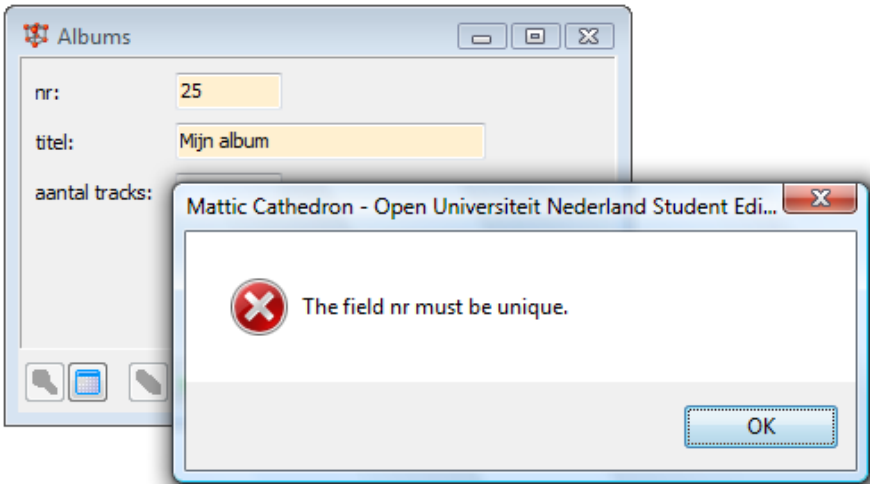
Als we voor Album een specifiek nr, titel of aantal tracks kennen, kunnen we deze direct invullen in het zoekvenster. Er zijn dan drie mogelijkheden:

1 De zoekoperatie levert geen resultaten op. Cathedron meldt dit.
2 De zoekoperatie levert één record op. Het oorspronkelijke formulier (in dit geval Album) wordt geopend met de gegevens van het gevonden record. In geval van een master-detailformulier worden ook de gegevens van de bijbehorende details getoond.
3 De zoekoperatie levert meerdere records op. Nu toont Cathedron een formulier met de gegevens van records die voldoen aan het zoekcriterium en geeft het de mogelijkheid hier één record uit te selecteren of alle records. Bij selectie van één record gebeurt hetzelfde als bij 2. Bij selectie van alle records wordt het oorspronkelijke formulier getoond in mrv en fungeert de zoekoperatie als een filter: met de navigatieknoppen kunt u door de tabel bladeren en krijgt u alleen records te zien die voldoen aan het zoekcriterium.

Open Muziekcollectie1 met Cathedron.
a Zoek de gegevens op van Album met nr = 10.
b Zoek het album Brothers In Arms. Wijzig de titel van de zevende track in 'The Man Is Too Strong' en post deze wijziging.
c Voeg de gegevens van een eigen album met bijbehorende tracks toe.
d Probeer een track van een album te verwijderen. Probeer een album te verwijderen. Wat valt u op ? Is daar een verklaring voor?
e Probeer het nummer van album 19 te wijzigen in 22. Waarom lukt dit niet?
f Wijzig het volgnr van track 13 op album 14 in 23. Wat valt u op?

Ad c

In figuur 8 ziet u wat er gebeurt wanneer u een albumnummer (nr) probeert in te voeren dat al gebruikt wordt. In dat geval zal de post niet slagen.



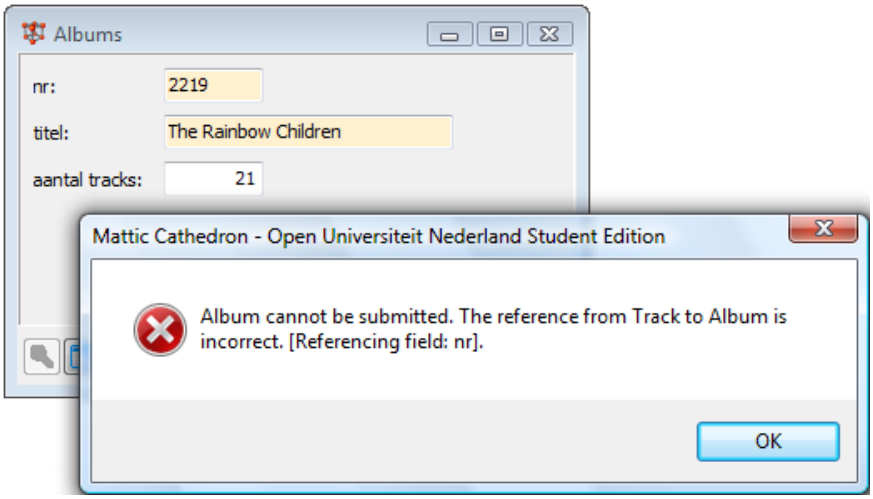
FIGUUR 8 Identificerend attribuut nr moet uniek zijn

Ad d

Het verwijderen van een album gaat niet. Dit komt omdat er anders tracks zouden zijn zonder album en volgens het informatiemodel kan dat niet. Eerst moeten dus alle tracks van een album worden verwijderd. (Cathedron ondersteunt geen cascading delete, zie paragraaf 2.4.)

Ad e

Het nummer van album 19 wijzigen lukt niet. Dit komt omdat alle tracks van het album nog een verwijzing hebben naar het oude nr (zie figuur 9). Conclusie: de verwijzing van Track naar Album heeft geen 'cascading update'.



FIGUUR 9 Geen 'cascading update': nr niet te wijzigen

Ad f

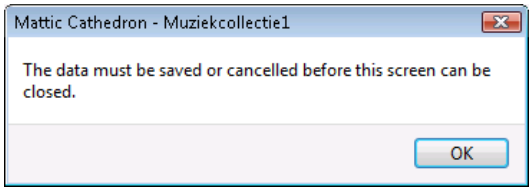
Een trackvolgnr wijzigen lukt wel, maar de tracks worden niet automatisch gesorteerd op tracknummer (volgnr).

Open formulier Album, geef als zoekcriterium voor nr not empty, doe een search en kies voor Select All.
a Ga na dat nu automatisch een master-detailformulier geopend wordt met de master in mrv.

b Klik op de knop voor single record view. Probeer daarna de navigatietoetsen uit.
c Klik ook eens op de knop single record view in het detailvenster.
d Open form Album opnieuw maar zorg nu dat Auto create details is uitgeschakeld. Ga na dat Cathedron nu geen master-detailformulier toont maar alleen het formulier van Album in srv.
e Klik op knop edit en probeer in editmode het venster te sluiten door op het kruisje in de titelbalk van het formulier te klikken.

Ad e

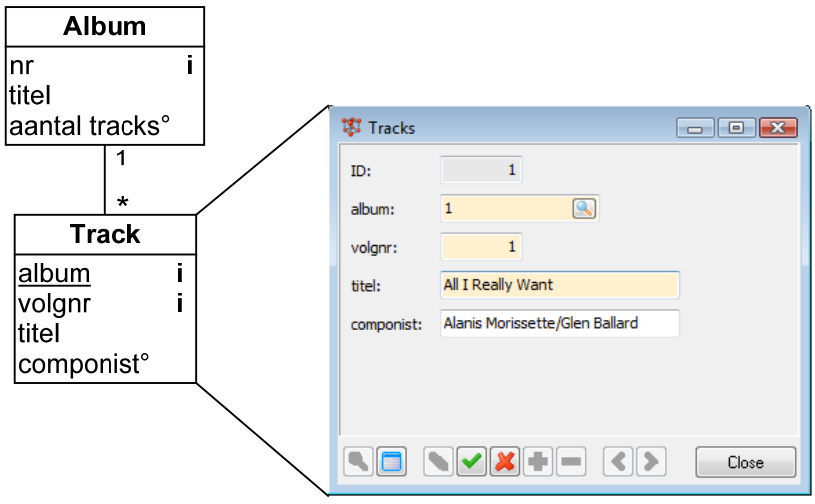
Cathedron staat niet toe een venster te sluiten zonder de transactie expliciet af te sluiten. U moet expliciet op één van de knoppen Cancel of Post klikken voordat u het venster kunt sluiten (zie figuur 10).



FIGUUR 10 Waarschuwingsvenster van Cathedron

Detailformulieren
We hebben gezien dat Cathedron in geval van een 1-op-n-associatie bij het openen van een formulier dat correspondeert met de 1-kant, een master-detailformulier opent. We vragen ons nu af wat we te zien krijgen als we een formulier aan de n-kant openen.

We sluiten eventueel geopende formulieren en openen formulier Track, zie figuur 11.



FIGUUR 11 Trackformulier

We zien een extra label ID: het veld correspondeert met een databasekolom Track.id. Deze kolom is een kunstmatige sleutel, die wordt gegenereerd wanneer de klasse geen enkelvoudige identificatieregel heeft. Kolom Track.id wordt gerealiseerd met een automatische nummergenerator, waardoor u dit nummer niet zelf kunt

invoeren of wijzigen. Cathedron toont voor dit attribuut een grijze achtergrond ten teken dat de waarde niet editbaar is.
In dit venster is kolom album zichtbaar met daarin het nr van het album waar deze track bijhoort. Deze kolom is niet zichtbaar in het Track-frame binnen het master-detailformulier van Album.

Aanpassing van formulieren wordt behandeld in blok 2.

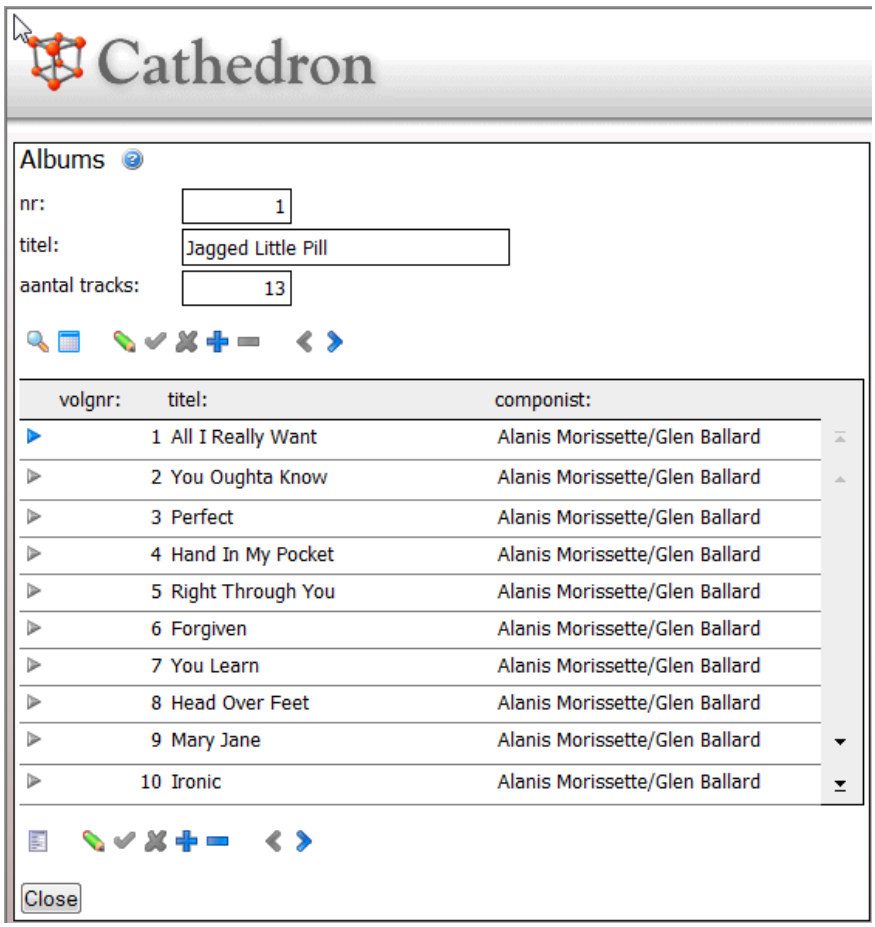
Dit op zichzelf staande Track-formulier is niet echt nuttig voor gewone gebruikers. In de definitieve applicatie zal het voor hen waarschijnlijk niet toegankelijk zijn. Overigens is het formulier volledig aanpasbaar, zodat bijvoorbeeld de Track.id's niet worden afgebeeld en naast de albummummers ook de albumtitels worden getoond.

Als een klasse een n-op-1-associatie heeft met een andere klasse (ouderklasse), biedt het formulier een zoekmogelijkheid, via een knopje met een vergrootglas. In figuur 11 zien we zo'n knopje naast kolom album. Hiermee kunnen we in edit- of insertmode een zoekformulier voor de ouder openen en in dit geval zoeken naar een album waarvan we het nummer moeten invoeren bij een track.

Webinterface
Het model Muziekcollectie1 is neutraal ten opzichte van het platform waarop de applicatie draait. We hebben gezien hoe Muziekcollectie1 geopend kan worden met een Windows-GUI. We zullen nu laten zien hoe eenvoudig het is Muziekcollectie1 te openen met een webinterface.

Open de Cathedron Manager en selecteer Muziekcollectie1.
Druk op de knop Run WebCat. Start daarna een webserver op via de knop Start Server. Druk tot slot op Open in browser.
Log in met dev, dev.
Er wordt nu een menu geopend voor de 'Default main forms' waarin u kunt kiezen tussen de formulieren Album en Track.
Figuur 12 toont het resultaat wanneer we voor Album kiezen.

Vergeet niet de server te stoppen wanneer u weer wilt overschakelen naar een Windows GUI.



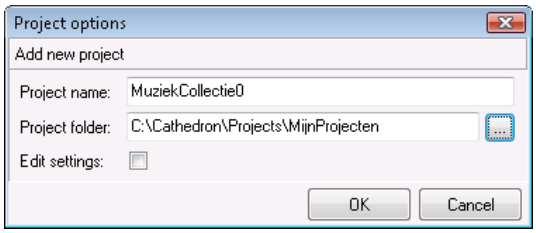
FIGUUR 12 Master-detailformulier van Album in webinterface

Bijlage 3

Cathedron als ontwikkelomgeving (practicum)

In deze bijlage stappen we in de rol van ontwikkelaar. In die rol bouwen we Muziekcollectie1 na om vertrouwd te raken met Cathedron als ontwikkelomgeving en ter illustratie van de theorie.

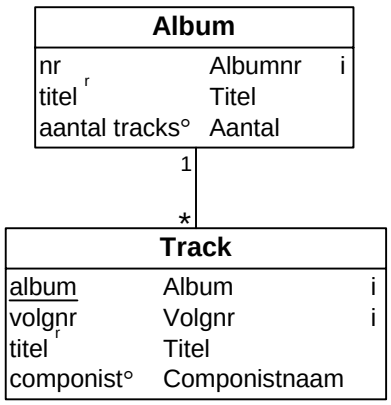
Nieuw project
Start Cathedron Manager zonder voorbeelden op, maak een nieuw project aan met knop New. Noem het project Muziekcollectie0 en maak een nieuwe map MijnProjecten aan in directory Projects van Cathedron door op het knopje naast 'Project folder:' te klikken. Kies de nieuwe map als Project folder (zie figuur 1). Druk op OK. Na inloggen met DEV/dev wordt Cathedron gestart met het nieuwe project.



FIGUUR 1 Een nieuw project Muziekcollectie0

De locatie van de Projects directory is afhankelijk van wat u tijdens de installatie van Cathedron hebt gekozen en kan daarom afwijken van de locatie in figuur 1.

Opbouw ontwikkelomgeving
Het informatiemodel is de kern van een MDD-model. Daar beginnen we dus mee. Omdat het doel is een applicatie te genereren, moet het model voldoende gespecificeerd zijn. Dat betekent dat we van ieder attribuut van een klasse ook het attribuuttype moeten specificeren. In figuur 2 herhalen we het informatiediagram van figuur 1 van bijlage 1. De kern van onze ontwikkeltaak bestaat erin dit model in te voeren in de ontwikkelomgeving.



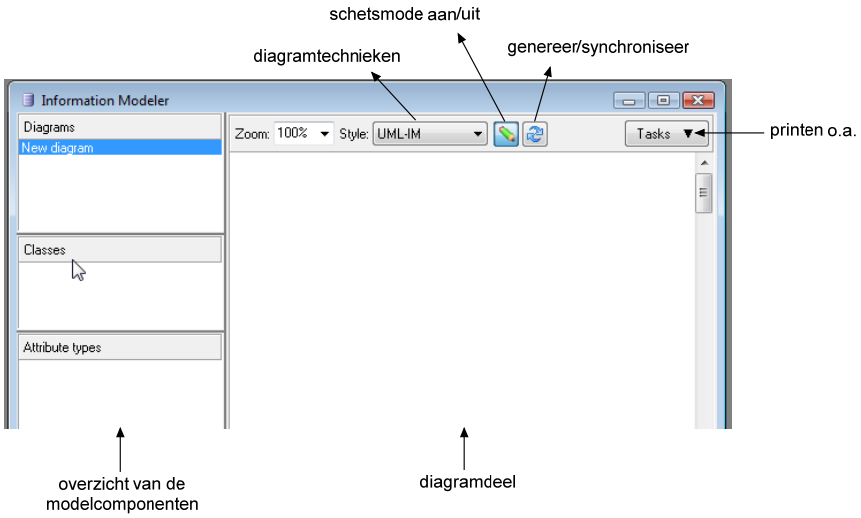
Attribuuttype

naam	*datatype*
Albumnr	Integer
Titel	Varchar (100)
Aantal	Integer
Album	(klasse)
Volgnr	Integer
Componistnaam	Varchar (50)

FIGUUR 2 Informatiediagram (volledig informatiemodel) van Muziekcollectie0

Information Modeler

Het informatiemodel wordt in Cathedron ingevoerd via de 'Information Modeler', zie figuur 3. Start de Information Modeler via Design | Information Modeler.



FIGUUR 3 Information Modeler van Cathedron bij nieuw model

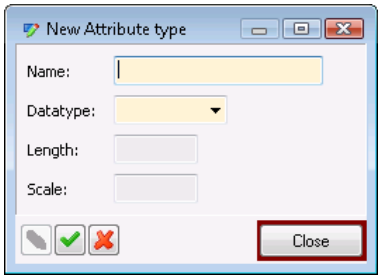
Schetsmode en uitgebreide mode

In deze bijlage gebruiken we de Information Modeler alleen in de *schetsmode* (te zien aan het ingedrukte potloodknopje bovenin). In de schetsmode laten de invulformulieren alleen de hoognodige velden zien. In het begin is dat erg prettig. Later zult u meer velden nodig hebben en schakelt u over naar de wat complexere formulieren van de *uitgebreide mode*.

Attribuuttypen

Omdat alle attribuuttypen bekend zijn (zie figuur 2) is de eenvoudigste werkwijze om alle attribuuttypen vooraf te creëren, volgens we het eerste stappenplan van paragraaf 2.9. De attribuuttypen komen dan, bij de creatie van attributen, beschikbaar in een keuzelijst. Desgewenst kunt u de attribuuttypen of een deel ervan later 'just in time' aanmaken, zoals verderop beschreven in de paragraaf Klassen en attributen.

Voor creatie vooraf van een attribuuttype klikt u in vak Attribute types met de rechtermuisknop en selecteert u New. Er wordt dan het formulier van figuur 4 geopend.



FIGUUR 4 Formulier voor de invoer van een nieuw attribuuttype

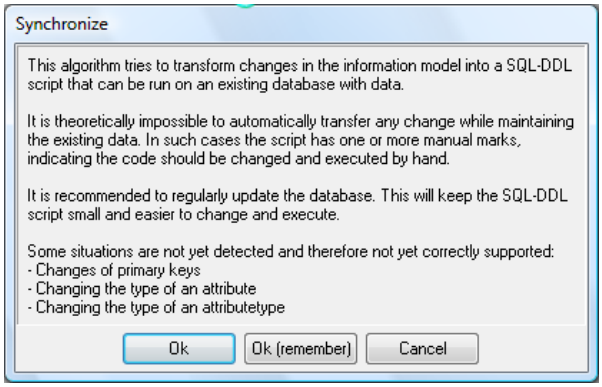
De afrollijst bij Datatype: toont alle beschikbare standaard datatypen van Firebird. Voor sommige daarvan is de lengte nodig en voor type decimal zelfs lengte en schaal.

Voer nu de attribuuttypen in van figuur 2.

Genereren / synchroniseren
Na het invoeren van de attribuuttypen kunnen de corresponderende databaseobjecten worden gegenereerd: domeinen met hun datatypen. We hoeven hiermee niet te wachten tot we ook de klassen met hun attributen hebben ingevoerd.
Elke toekomstige wijziging of aanvulling van het informatiemodel leidt tot synchronisatie van de onderliggende relationele database. De synchronisatie wordt uitgevoerd door onder andere een SQL-script, dat bekeken en eventueel veranderd kan worden voor het wordt uitgevoerd.

Genereer nu de databaseobjecten (domeinen) bij de ingevoerde attribuuttypen. Druk hiertoe op de generatie/synchronisatieknop (zie figuur 3).

Cathedron toont nu een venster met enige uitleg en goede raad over het synchronisatieproces. Zie figuur 5. Het is belangrijk dit goed door te lezen.



FIGUUR 5 Waarschuwingen bij synchronisatie

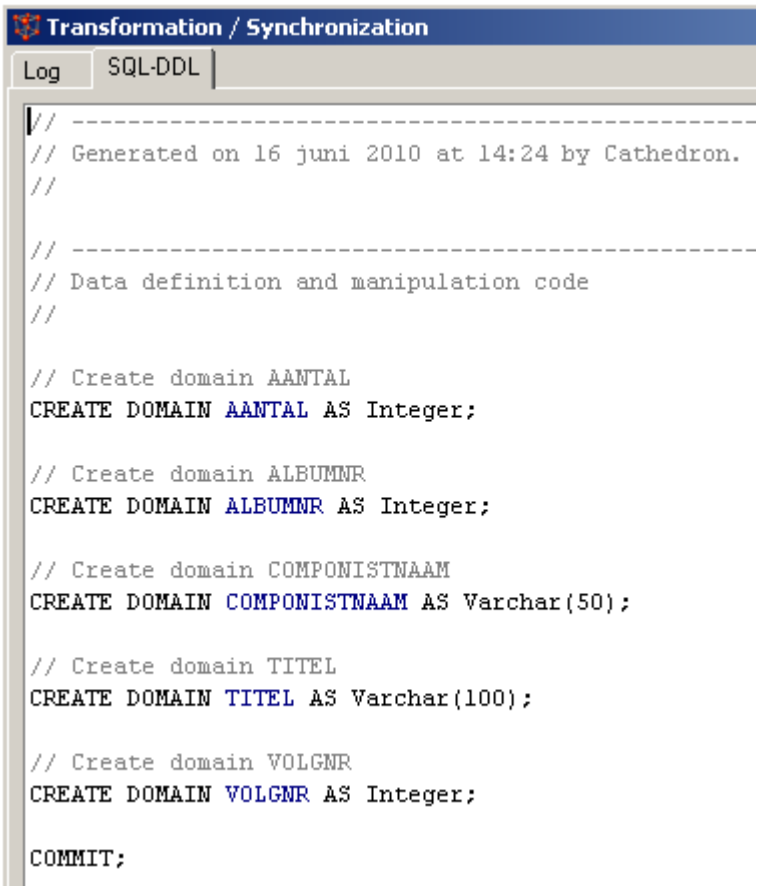
Voor dit moment is het belangrijkste punt dat regelmatig gesynchroniseerd moet worden (‘update the database’). Op de andere punten komen we verderop terug.

Het generatieproces levert een SQL-script op dat bekeken kan worden en vervolgens uitgevoerd. Het script zelf wordt opgeslagen, zodanig dat het ook later nog bestudeerd kan worden.

Druk op Ok. (Ok-remember zorgt dat dit venster niet terugkeert; dat is iets voor later.)
Bekijk vervolgens het SQL-script (zie figuur 6) en laat dit uitvoeren met Execute Script.

Voor domeinen: zie cursus Databases

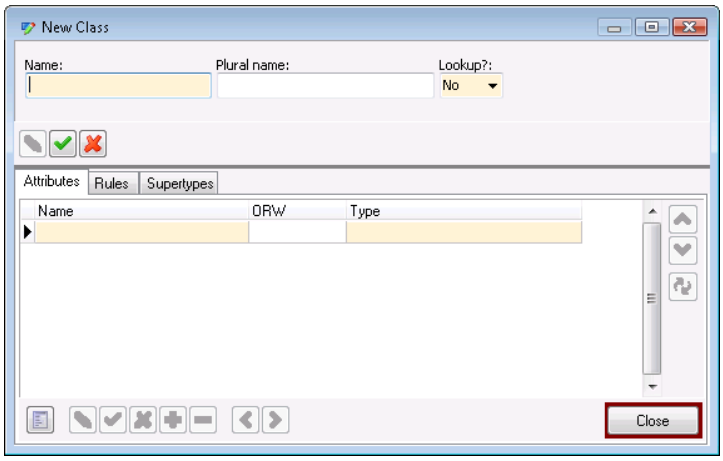
Er zijn nu binnen de relationele database vijf domeinen aangemaakt, elk met een bijbehorende datatype. In een volgende stap zullen we aanstonds aan elk attribuut een domein toewijzen en via dat domein dus ook een datatype.



FIGUUR 6 SQL-DDL-tabblad met gegenereerd SQL-script

Klassen en attributen
We kunnen nu de klassen Album en Track creëren met hun attributen. Een nieuwe klasse kunnen we aanmaken door in het onderdeel Classes (links in de Information Modeler) of in het diagramdeel (rechts) met de

rechtermuisknop te klikken. In beide gevallen wordt een master-detailformulier New Class geopend, zie figuur 7.



FIGUUR 7 Master-detailformulier New Class

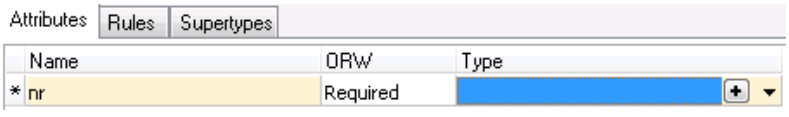
In het masterdeel kunnen we de volgende eigenschappen van de klasse invoeren:
– bij Name geven we de klassenaam op (Album, Track).
– Bij Plural name geven we de meervoudsvorm op (Albums, Tracks); deze wordt gebruikt als default titel voor het formulier.

Lookupklasse

– met Lookup kunnen we vastleggen dat een klasse als *lookupklasse* behandeld moet worden. Dat is een klasse voor (standaard)gegevens die beschikbaar zijn als keuzelijst in een formulier van een andere klasse. Een voorbeeld van een lookupklasse is een klasse met alle plaatsnamen van Nederland. We komen terug op lookupklassen in leereenheid 8 'Gebruikersinterface'.

Merk op dat de meervoudsvorm geen onderdeel is van het informatiemodel, maar van de interfacespecificatie! We zullen nog vaker zien dat we bij het informatiemodel 'en passant' interface-elementen of zelfs eenvoudige bedrijfsregels specificeren.

Het detailgedeelte bevat drie tabbladen. Voor ons is nu alleen het tabblad Attributes van belang:
– bij Name geven we de attribuutnaam op.
– onder ORW ('optional, required, warning') geven we aan of een attribuut optioneel of verplicht is. De waarde 'warning' leidt tot een optionele kolom in de database, maar de gebruiker krijgt een waarschuwing wanneer geen waarde wordt ingevuld.
– onder Type kunnen we het attribuuttype opgeven. Als we bijvoorbeeld bij klasse Album attribuut nr invoeren, moeten we een attribuuttype voor nr aangeven, zie figuur 8.



FIGUUR 8 Attributes-tabblad met attribuut nr

Als u op de pijl klikt, kunt u een attribuuttype kiezen uit de al door u gemaakte attribuuttypen en klassen. (Dit is een voorbeeld van een lookuplijst.) Als u op de + klikt, kunt u een nieuw attribuuttype (of klasse) toevoegen. Dit is de 'just in time'-definitie waar we eerder over spraken in paragraaf 2.6.

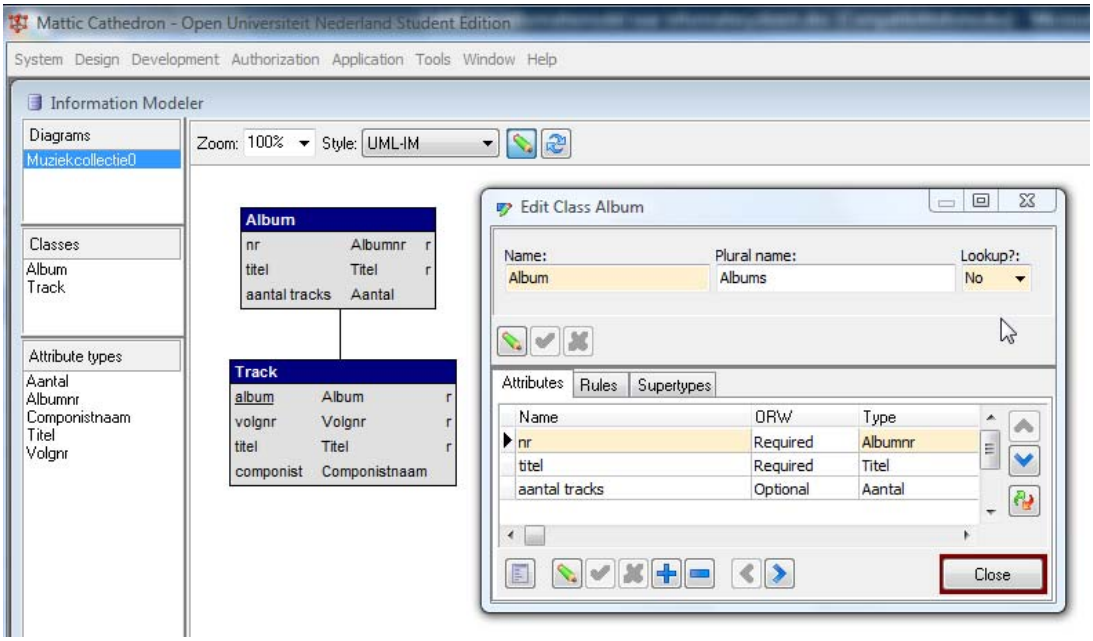
Attribuuttypen omvatten ook klassen!

Merk op dat de attribuuttypelijst ook de klassen bevat. Bedenk hierbij dat het attribuuttype van een associatieattribuut een klasse is. Op deze manier kunt u dus eenvoudig een associatie tussen twee klassen realiseren. Zo zal klasse Track een attribuut album van het type Album krijgen. In het informatiediagram wordt zo'n attribuut onderstreept en er wordt een lijntje getekend naar de betreffende tabel. We tekenen dus niet zelf de verbinding tussen de verschillende klassen. (De Information Modeler is geen tekenprogramma, al kunnen we een getekende lijn wat verschuiven of van een knik voorzien.)

Synchronisatie pas na invoeren van identificatieregel!

We raden u aan de database pas te synchroniseren met een nieuwe klasse nadat u een weldoordachte identificatieregel hebt toegevoegd. Zie hiervoor de volgende paragraaf. Synchroniseert u per ongeluk eerder of gaat er anderszins iets fout met de synchronisatie, dan kunt u te allen tijde eerdere synchronisaties ongedaan maken door de optie Clear PSM and database onder het Tasks-menu.

Voer de klasse Album in op basis van figuur 2. Klik hiertoe met de rechtermuisknop in het vak Classes en kies New class.
Voer evenzo de klasse Track in. Kijk of het resultaat overeenkomt met figuur 9. Breng eventuele wijzigingen aan met Edit (rechtermuisknop).
Wacht nog met genereren/synchroniseren, totdat u – in de volgende paragraaf – de identificatieregels hebt gespecificeerd.



FIGUUR 9 Klassendiagram in Information Modeler met editformulier voor klasse Album

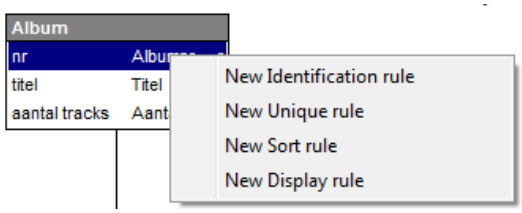
Opmerkingen:
– De naam van een diagram kunt u wijzigen via de optie Rename, na selectie met de rechtermuisknop.
– Bij één model kunt u meerdere diagrammen maken, die elk een deel van het model weergeven. Bij grote modellen (tientallen of honderden klassen) kan dit prettig zijn. De diagrammen zijn onafhankelijk van elkaar. U bepaalt zelf bij elk diagram welk deel van het model u wilt zien.

Identificatieregels

Hoewel Cathedron in staat is een default applicatie te genereren op basis van de tot nu toe ingevoerde gegevens, is het model nog niet compleet. We moeten voor elke klasse nog aangeven welk attribuut of welke attribuutcombinatie identificerend is.

Manier 1 om een identificatieregel aan te geven.

Dat kan op twee manieren. De eerste manier werkt met aanklikken in het diagram. Als er één identificerend attribuut is, zoals bij Album, selecteert u dat door de shifttoets ingedrukt te houden en met de muis op het attribuut te klikken. Daarna kiest u met de rechtermuisknop New Identification rule, zie figuur 10.

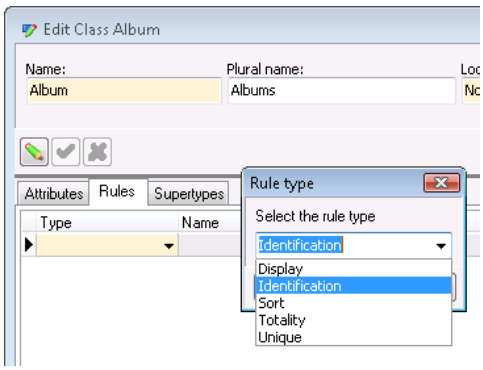


FIGUUR 10 New Identification rule

Als u een attribuutcombinatie identificerend is, zoals bij Track, moet u de shifttoets én de ctrl-toets ingedrukt houden en met de muis de gewenste attributen selecteren. Daarna kiest u weer New Identification rule met de rechtermuisknop.

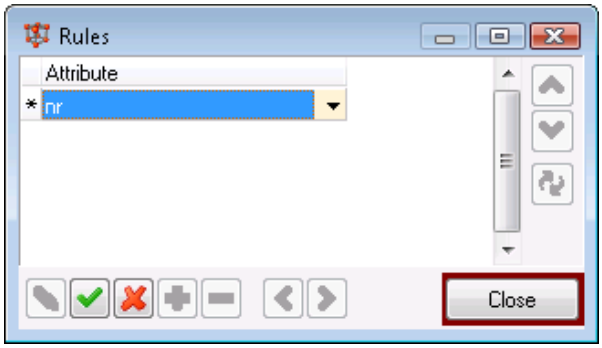
Manier 2 om een identificatieregel aan te geven.

Een tweede manier gaat via het invoer-/editformulier van de geselecteerde klasse (rechtermuisknopmenu). In het detailgedeelte klikken we op tabblad Rules en daarin voegen we via de +-knop een nieuwe rule toe, zie figuur 11.



FIGUUR 11 Een nieuwe rule toevoegen

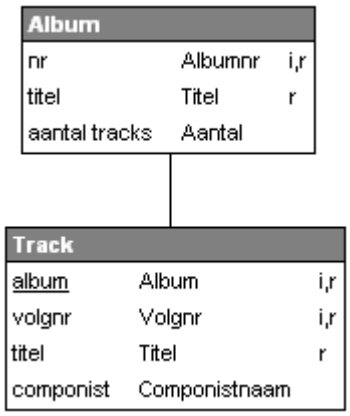
We kiezen voor Identification. Daarna wordt een formulier Rules geopend waarin we een attribuut kunnen aangeven dat identificerend moet zijn of dat onderdeel is van een identificerende attribuutcombinatie. Zie figuur 12, waarin we attribuut nr als identificerend hebben aangegeven.



FIGUUR 12 nr als identificerend attribuut

Na een post kunnen we eventueel nog een attribuut toevoegen in het geval van een identificerende attribuutcombinatie. Als we klaar zijn, kunnen we met Close het formulier sluiten. Na sluiten van het Class-formulier kunt u in het diagram zien dat achter het identificerende attribuut een i verschenen is. Bij een identificerende combinatie heeft elk attribuut een i.

Voer de identificatieregels in voor Album en voor Track. Zolang u niet synchroniseert, kunt u experimenteren, bijvoorbeeld door de regels te verwijderen en op een andere manier (manier 1 of manier 2) opnieuw in te voeren.
Als u tevreden bent en uw model overeenkomt met figuur 13, drukt u op de synchronisatieknop, bekijkt het SQL-script en voert dit uit (Execute script).



FIGUUR 13 Klassendiagram met identificatieregels in Information Modeler

Testen en wijzigen van de applicatie
Door de synchronisatiestap zijn niet alleen databasetabellen gegenereerd, maar er is ook een default applicatie beschikbaar. We kunnen deze nu gaan onderzoeken en uittesten.

Open het master-detailformulier van Album (zie bijlage 1 voor de verschillende manieren).
Voer één of meer albums in, elk met enige tracks. Test de datatypen, door invoer die daarmee in strijd is (bijvoorbeeld een te lange componistnaam of een non-integer op een plek waar een integer verwacht wordt. Test de identificatieregels, door invoer van een album met een al bestaand nummer of een track met een al bestaand volgnr op hetzelfde album.
Merk op dat de waarde van het attribuut 'aantal tracks' door uzelf ingevoerd moet worden. Zoals gezegd wordt dit later een afleidbaar attribuut, waarvan de waarde automatisch wordt berekend.