Ontwikkeling van een gepersonaliseerde webtool voor de predictie van aarfusarium

Jan Vandepitte

Promotoren: prof. Peter De Neve, prof. dr. Bernard De Baets
Begeleiders: Sofie Landschoot, Wim Van Lancker

Masterproef ingediend tot het behalen van de academische graad van
Master in de toegepaste informatica

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. Jan Van Campenhout

Vakgroep Toegepaste Wiskunde, Biometrie en Procesregeling
Voorzitter: prof. dr. Jean-Pierre Ottoy

Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2010-2011

UNIVERSITEIT
GENT

Gent, Augustus 2011

De promotoren

Prof. Peter De Neve

Prof. dr. Bernard De Baets

De auteur

Jan Vandepitte

Woord vooraf

Met dank aan mijn promotoren Prof. Peter De Neve voor het aannemen van mijn onderwerp en Prof. dr. Bernard De Baets om mij toe te laten verder bij te dragen aan het onderzoek van mijn eerste masterproef. Ook dank aan mijn tutors Ir. Sofie Landschoot en Ir. Wim Van Lancker voor het vrijmaken van hun tijd voor mijn begeleiding. Daarnaast ook dank aan Dr. Ir. Annemie Elsen, Patrick Van Damme en Wim Robijns van de Bodemkundige Dienst van België voor het beschikbaar stellen van hun ontwikkelingsomgeving en het beantwoorden van mijn technische vragen. Uiteindelijk ook dank aan mijn ouders om mij toe te laten een bijkomend jaar verder te studeren.

Jan Vandepitte
Gent, augustus 2011

Samenvatting en sleutelwoorden

Samenvatting

In deze masterproef wordt de basis gelegd voor een beslissingondersteunende webtool in dienst van de problematiek van *Fusarium* en de daarbij geproduceerde mycotoxines op wintertarwe. Hiervoor werd eerst een analyse uitgevoerd van de gewenste functionaliteit en de reeds bestaande systemen en databanken waarop de webtool zou worden gebaseerd. Daarna volgde het technische ontwerp en de eigenlijke implementatie van de webtool. Uiteindelijk werd deze masterproef geschreven als documentatie voor de webtool.

Sleutelwoorden

Fusarium,DON, UML, ER, PL/SQL, *Oracle Application Express* (APEX), AnyChart, *stored procedure*, *trigger*, beslissingondersteunend

Inhoudsopgave

Lijst met afkortingen

Afkorting	Voluit
APEX	*Oracle Application Express*
ASP	*Active Server Pages*
BDB	Bodemkundige Dienst van België
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
DON	DeOxyNivalenol
ER	*Entity Relationship*
HTML5	*HyperText Markup Language 5*
JDBC	*Java DataBase Connectivity*
KMI	Koninklijk Meteorologisch Instituut van België
LCG	Landbouwcentrum Granen
MAP	MestActiePlan
NIS	Nationaal Instituut voor de Statistiek
OCI	*Oracle Call Interface*
OMT	*Object-modeling technique*
OOSE	*Object-Oriented Software Engineering*
PCA	Proefcentrum Aardappelenteelt
PHP	*PHP: Hypertext Preprocessor*
PL/SQL	*Procedural Language/Structured Query Language*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Call*
SQL	*Structured Query Language*
SQLJ	*Structured Query Language Java*
SWF	*Small Web Format*
UML	*Unified modelling language*
XML	*Extensible Markup Language*

Hoofdstuk 1

Inleiding

De problematiek van schimmelziekte *Fusarium* op wintertarwe heeft vele dimensies. Deze schimmel overleeft tijdens de winter in de bodem en op gewasresten van de voorvrucht die achterblijven op de akker. Bij deze overlevingsstructuren kan onderscheid gemaakt worden tussen de seksuele en asekseuele sporen (hyfe-elementen). Bij de seksuele sporen kan er ook nog eens onderscheid tussen drie verschillende sporen. Enerzijds maakt men onderscheid tussen ascosporen en macronidia. Anderzijds kunnen de macronidia onder invloed van slechte weeromstandigheden omgevormd worden tot overlevingsstructuren door verdikking van de celwand (zogenaamde chlamydosporen). Tevens beïnvloedt het weer de ontwikkeling van het tarwegewas. De tarweplant heeft behoefte aan een hoeveelheid koude dagen om aan zijn vernalisatiebehoefte te voldoen. Bij de vernalisatie wordt het topmeristeem die later de aar zal vormen voorbereid om de overgang te maken van de vegetatieve naar de reproductieve fase. Daarnaast is wintertarwe een kwantitatieve lange-dag-plant, wat betekent dat de reproductie vroeger start naarmate de duur van zonneschijn langer wordt. De macroconidia kunnen de tarweplant over het gehele groeiseizoen infecteren en koloniseren terwijl het vruchtlichaam dat de ascosporen produceert onder invloed van het weer tot rijping moet komen om zijn sporen te kunnen vrijstellen. Tijdens de bloei van tarwe is de plant extra gevoelig voor infectie doordat meeldraden een stof produceren die de groei van de schimmel stimuleren. De koloniseerbaarheid van de aar is ook afhankelijk van in welk rijpingsstadium dat de aar zich bevindt. De schimmelziekte kan als hij de aar heeft gekoloniseerd ook giftige secundaire metabolieten produceren, de zogenaamde mycotoxines. Een van de belangrijkste hiervan is deoxynivalenol (DON). Deze secundaire metabolieten worden geproduceerd onder invloed van bepaalde stressfactoren. Het kan hier gaan om ongunstig weer maar ook om een behandeling met een bepaalde klasse van fungiciden. Van aarfusarium zijn er ook verschillende species die niet noodzakelijk allemaal deze mycotoxines produceren. De aanwezigheid van deze species kan ook sterk verschillen van regio tot regio. Hierdoor is de kolonisatie van de aar door de schimmel zeer moeilijk te correleren met de productie van de mycotoxines. Daarnaast zijn er bij wintertarwe ook verschillen tussen de variëteiten bijvoorbeeld in de aanwezigheid van bepaalde fysiologische resistentiereacties op de infectie of de plantenarchitectuur die de kolonisatie kan beïnvloeden. De mycotoxines vormen een gezondheidsrisico en voedselproducenten en landbouwers hebben wettelijke verplichtingen om te verhinderen dat deze in de voedselketen terecht komen. Door de sterke lokale en seizoenale effecten op de ontwikkeling van deze plaag is het noodzakelijk dat de akkerbouwer ondersteund wordt in de behandelingen hiertegen. Het doel van de webtool, is een gebruiker te ondersteunen met een informatie over deze problematiek. Met een persoonlijk profiel waarin de voorgeschiedenis van percelen kan bijgehouden worden en een voorspellingsmodel op perceelniveau wordt de gebruiker verder ondersteund.

Hoofdstuk 2

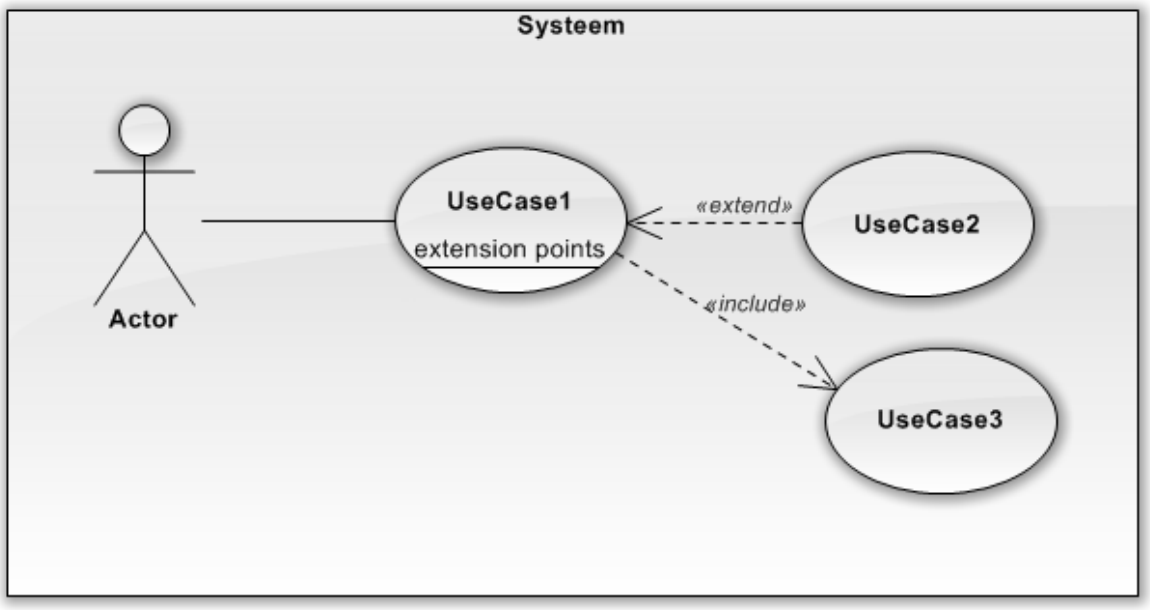
Concepten en achtergrond van gebruikte technologieën

2.1 UML en ER

Het doel van *Unified Modelling Language* (UML) is om software ontwikkelaars te voorzien van hulpmiddelen voor analyse, ontwerp en implementatie van software-gebaseerde systemen. Het kan bij uitbreiding ook gebruikt worden voor het modelleren van bedrijfsprocessen of gelijkaardige processen. De initiële versie van UML was voornamelijk gebaseerd op drie van de belangrijkste objectgeoriënteerde methoden (Booch, *Object-modeling technique* (OMT) en *Object-Oriented Software Engineering* (OOSE)). Het hoofddoel van UML is om interoperabiliteit tussen visuele modelleringshulpmiddelen te verbeteren.

In deze thesis wordt gebruikt gemaakt van *use case*-diagrammen (zie Figuur 2.1) als hulpmiddel in de analysefase om de vereiste functionaliteit van de software vast te leggen. Hierbij worden de verschillende actoren die met de software interageren geïdentificeerd alsook de functionaliteiten (de zogenaamde *use cases*) die voor deze actoren gewenst zijn (4). Bij het identificeren van actoren, kan men als maatstaf nemen dat men het gedrag van deze entiteiten niet kan controleren of veranderen. Het gaat hier dus meestal om de menselijke gebruikers, maar het kan dus evengoed om systemen gaan zoals servers en databanken onder controle van andere personen of *legacy*-systemen. De verschillende *use cases* kunnen gegroepeerd worden in systemen. Daarnaast kunnen er ook pijlen getekend worden tussen de actoren en zijn respectievelijke functionaliteit of tussen de functionaliteiten zelf. In het tweede geval kan het hier dan gaan over een functionaliteit die andere functionaliteit hergebruikt (<<includes>>) of uitbreidt (<<extends>>).

Voor de analyse van de bestaande omgeving en het technische ontwerp van de software werden enerzijds *entity-relationship*-diagrammen en anderzijds een *deployement*-diagram opgesteld. De *entity-relationship*-diagrammen (ER) laten toe de entiteiten en hun onderlinge relaties van een logisch datamodel of de tabellen en sleutels van een relationeel datamodel voor te stellen. Bij de analyse van de bestaande omgeving kan deze dus gebruikt worden om de reeds bestaande databanken voor te stellen. Aan dit logisch datamodel konden dan in de ontwerpfase de entiteiten en relaties toegevoegd worden die geïdentificeerd waren bij de functionele analyse. Op basis van deze ER-diagrammmen kan nieuwe datadefinitietaal (DDL) voor een specifiek databanksysteem gegenereerd worden. De diagrammen kunnen omgekeerd gegenereerd worden uit DDL.

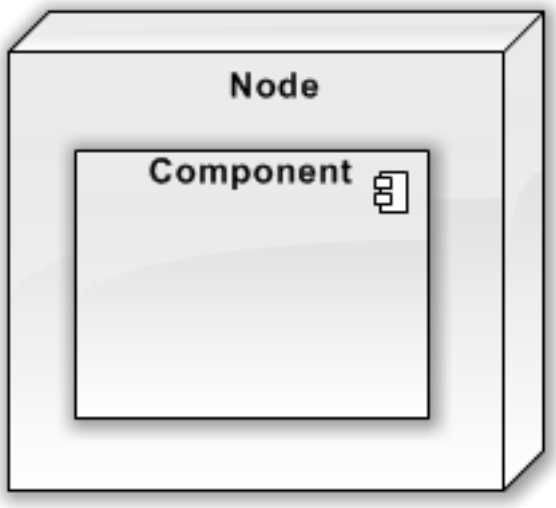


Figuur 2.1: Een voorbeeld van hoe een actor en *use cases* worden voorgesteld in een *use case*-diagram.

Het *deployement*-diagram (zie Figuur 2.2) laat toe de aanwezige hardware (*nodes*) en geïmplementeerde software (*components*) te beschrijven en daarbij ook te beschrijven hoe nieuwe hardware en software te implementeren. Aangezien uit de analyse van de bestaande omgeving bleek dat de webtool geen implementatie vereist met een objectgeörienteerde taal (bij de ontwikkeling van de webtool werden procedures geschreven in PL/SQL (*Procedural Language/Structured Query Language*)), werden geen diagrammen opgesteld die klassen en interacties hiertussen voorstellen en worden deze ook niet behandeld in deze thesis.

2.2 Oracle DB en PL/SQL

In databanksystemen wordt via een *select*-uitdrukking een subset van *records* en kolommen van één of meerdere tabellen opgevraagd. Deze resultaatset van deze *query* kan dan rechtstreeks gebruikt worden in de applicatie die bovenop het databanksysteem gebouwd is. In functie van de overzichtelijkheid en abstrahering van de SQL-uitdrukking kan er gekozen worden om met *views* te werken. *Views* zijn databankobjecten die gebruikt worden om een *select*-uitdrukking op te slaan. Een *view* kan dan in andere *select*-uitdrukkingen of *views* gebruikt worden alsof ze een tabel met de resultaatset bevat van de *select*-uitdrukking die de *view* voorstelt zonder dat deze fysiek in het databanksysteem moet opgeslagen zijn. In het Oracle-databanksysteem bestaat de mogelijkheid om deze resultaatset wel te materialiseren op de databank, zogenaamd als een *materialized view*. Dit kan nuttig zijn als de *select*-uitdrukking veel *join*-operaties of aggregerende functies bevat, aangezien bij het frequent opvragen van de *view* de achterliggende berekeningen en optimalisaties niet telkens moeten uitgevoerd worden door het databanksysteem. Hierdoor is de integriteit van de achterliggende primaire en vreemde sleutels gevrijwaard. Aggregerende functies zijn functies die reeds in het databanksysteem ingebouwd zijn en dus niet te verwarren met PL/SQL-functies die de databankgebruiker zelf definieert. Deze geaggregeerde functies dienen om onder andere statistieken zoals percentielen te berekenen op kolommen of nulwaarden op te vangen (al dan niet per gegroepeerde subset van de tabel) (6).



Figuur 2.2: Een voorbeeld van hoe een node en een component worden voorgesteld in een *deployement*-diagram.

PL/SQL is een uitbreiding van Oracle op SQL om aspecten van procedurale programeertalen toe te voegen. Deze wordt gebruikt als programmeertaal in *triggers*, *constraints*, *stored procedures* en functies op de databank (11). Elk PL/SQL programma bestaat uit een combinatie van SQL en PL/SQL-*statements* die samen een PL/SQL-blok vormen. Deze kunnen dus opgeslagen worden als ondermeer *stored procedures* en functies of direct uitgevoerd worden (zogenaamde anonieme blokken). Alle statements binnen een blok moeten eindigen met een puntkomma. PL/SQL-blokken kunnen genest worden of opgeslagen procedures aanroepen. Één blok kan bestaan uit drie secties:

- Een optionele declaratiesectie start met het gereserveerd sleutelwoord **DECLARE**. In deze sectie worden alle plaatshouders zoals variabelen, constante waarden, records en cursoren (voor programmalussen) vooraf gedeclareerd.
- De verplichte uitvoeringsectie met alle programmalogica geschreven in lussen, conditionele *statements* en SQL-*statements*. Het start met het sleutelwoord **BEGIN** en eindigt met het sleutelwoord **END**.
- Een optionele sectie voor de afhandeling van fouten en uitzonderingen kan zich in de uitvoeringssectie bevinden. Deze start met het sleutelwoord **EXCEPTION**. Als er fouten of uitzonderingen zijn in de loop van het programma die niet in deze sectie worden behandeld kan het programma vroegtijdig afgebroken worden.

De volledige PL/SQL blok wordt ook beeindigd met een puntkomma. Een PL/SQL-blok heeft dus de volgende vorm(3):

Broncode 2.1: PL/SQL-blok

```
DECLARE
    --Variabele declaratie
BEGIN
    --Uitvoering programma
EXCEPTION
    --Behandeling fouten en uitzonderingen
END;
```

De term plaatshouder houdt in dat er een deel van het werkgeheugen tijdelijk wordt gereserveerd voor de opslag en manipulatie van een waarde tijdens de uitvoering van een blok. Deze waarde kan een variabele, constante of record zijn. De syntax voor de declaratie van variabelen is als volgt:

```
naam_variabele  datatype  [NOT NULL := waarde ];
```

De declaratie bestaat dus uit:

- de naam van de variabele,
- het datatype van de variabele (number(n), varchar(n), long, blob, clob,...),
- een optionele aanduiding of de variabele NULL mag zijn,
- een optionele toekenning van de waarde,
- een puntkomma op het einde.

Toekenning van een waarde aan een variabele kan gebeuren op twee manieren:

- rechtstreekse toewijzing:

```
naam_variabele := waarde;
```

- op basis van record in een databankkolom:

```
SELECT naam_kolom
INTO naam_variabele
FROM naam_tabel
[WHERE conditie];
```

Een constante is gelijkaardig aan een variabele met het verschil dat de toekenning van de waarde verplicht bij de declaratie moet gebeuren. Constanten worden aangeduid met het gereserveerde sleutelwoord **CONSTANT** na de naam van de constante in de declaratie. Een record is een variabele waarvan de datastructuur is samengesteld uit de scalaire datatypes van gewone variabelen. Deze datastructuur kan gecreëerd worden door zelf datatypes te declareren, deze op te bouwen uit het datatype van bepaalde databankkolommen (naam_kolom\%**TYPE**) of de gehele datastructuur van een databanktabel te kopiëeren (naam_tabel\%**ROWTYPE**):

Broncode 2.2: Definitie type

```
TYPE naam_record_type IS RECORD

(naam_eerste_kol  kolom_datatype ,

naam_tweede_kol naam_kolom%TYPE,  ...);
```

Voor de toekenning van waarden aan de individuele kolommen van een record is de aanspreking op een gelijkaardige manier als kolommen van gewone databank tabellen, namelijk als tabel .kolom. Datatypes kunnen ook vooraf al vast gedefinieerd zijn in het databankschema. Dit kadert binnen het integreren van concepten van objectgeoriënteerde databanksystemen in de SQL3 standaard. Met de syntax **CREATE TYPE** objectnaam **AS** kan dan een objecttype, een SQLJ objecttype, een varray (een rij van eenzelfde datatype met variabele lengte), een genest tabeltype of een onvolledig objecttype aangemaakt worden(5). Met de syntax **CREATE TYPE BODY** objectnaam **IS** kunnen dan ook nog methoden onder de vorm van *stored procedures* of functies bij deze objecttypes gedefinieerd worden. Als PL/SQL-blokken genest worden zijn de variabelen uit de buitenste blok beschikbaar voor zowel de buitenste

als voor de dieper geneste blokken (zogenaamd globale variabelen). De variabelen gedeclareerd voor een blok genest in een andere blok zijn daarentegen niet beschikbaar voor de andere, hoger geneste blok (zogenaamd lokale variabelen).

De PL/SQL-programmeertaal bevat ook de concepten van andere programmeertalen zoals voorwaardelijke *statements* en voorwaardelijke lussen (gewone lus, *while*-lus of *for*-lus). Specifiek als programmeertaal binnen een databanksysteem is er ook het concept van cursoren. Deze worden gebruikt om tijdelijk gegevens die van de databank zijn afgehaald te bewaren en manipuleren. Een cursor kan meer dan één rij bevatten, maar kan er maar één per keer verwerken. Er zijn twee types cursoren:

- Impliciete cursor: Deze worden impliciet aangemaakt bij het uitvoeren van datamanipulatie-euitdrukkingen (DML) zoals **INSERT**, **UPDATE** en **DELETE**. Ze worden ook aangemaakt als een **SELECT**-*statement* maar één rij oplevert. Van deze cursor kunnen dan ook eigenschappen opgevraagd worden over de uitgevoerde uitdrukkingen zoals \%**FOUND**, \%**NOTFOUND** en \%**ROWCOUNT**.
- Expliciete cursor: Deze moet aangemaakt worden bij het uitvoeren van een **SELECT**-*statement* die meer dan één rij oplevert. De rij in de cursor die op dat moment wordt verwerkt wordt de huidige rij (*current row*) genoemd. Wanneer de volgende rij wordt opgevraagd (*fetch*) schuift de cursor dus één rij verder.

Bij het uitvoeren van procedures en functies kunnen parameters doorgegeven worden. Deze kunnen bij de declaraties van de procedures of functies volgens de volgende syntax gedefinieerd worden:

Broncode 2.3: Soorten parameters bij definitie van een *stored procedure*

```
CREATE [OR REPLACE] PROCEDURE naam_procedure
( naam_param1 IN datatype,
naam_param2 OUT datatype,
naam_param3 IN OUT datatype, ... )
```

Het kan hier dus gaan om drie soorten parameters afhankelijk van welke rechten de functie of de procedure heeft over de geheugenplaats van de variabele. Als er het sleutelwoord **IN** tussen de parameternaam en zijn datatype staat, heeft het PL/SQL-blok enkel rechten om de waarde van de parameter te lezen. Dit is ook het standaard parametertype en het sleutelwoord kan dus in principe weggelaten worden. Als de functie of procedure enkel schrijfrechten mag hebben voor de geheugenplaats van de parameter wordt dit aangeduid met het sleutelwoord **OUT**. Als er zowel lees- als schrijfrechten mogen zijn voor de parameter is er het sleutelwoord **IN OUT**.

Een *stored procedure* is een PL/SQL-blok dat opgeslagen is onder een bepaalde naam en één of meerdere taken uitvoert. Een procedure heeft een hoofding voor de definitie van de parameters en optioneel een *return* datatype. Een *stored procedure* kan, maar moet niet een waarde teruggeven. Na de hoofding is er het PL/SQL-blok met de code van de procedure. Deze is gelijkaardig aan een anonieme blok met het verschil dat het een naam heeft gekregen in functie van herhaald gebruik. De algemene syntax van een *stored procedure* is:

Broncode 2.4: Definitie van een *stored procedure*

```
CREATE [OR REPLACE] PROCEDURE naam_procedure [lijst met parameters]
IS
      Variabele  declaratie
BEGIN
```

```
    Uitvoering Programma
EXCEPTION
    Behandeling Fouten en Uitzonderingen
END;
```

Het sleutelwoord **DECLARE** dat gebruikt wordt om de start van de declaratiesectie aan te duiden in anonieme blokken wordt voor *stored procedures* vervangen door **IS**. Een procedure kan op twee manieren uitgevoerd worden namelijk:

- Vanop de SQL-prompt:

```
EXEC[UTE] naam_procedure;
```

- Binnen een andere procedure:

```
naam_procedure;
```

Een functie is gelijkaardig aan een *stored procedure* met het verschil dat een functie verplicht is om een waarde terug te geven. De *return* is dus niet langer optioneel:

Broncode 2.5: Definitie van een functie

```
CREATE [OR REPLACE] FUNCTION naam_functie [parameters]
RETURN return_datatype;
IS
--PL/SQL-blok
```

Een *trigger* is een PL/SQL-blok die automatisch wordt uitgevoerd wanneer een datamanipulatieoperatie zoals een *insert*, *delete* of *update* op een bepaalde tabel of *view* uitgevoerd wordt. Het aanmaken van een *trigger* gebeurt volgens deze syntax:

Broncode 2.6: Definitie van een *trigger*

```
CREATE [OR REPLACE ] TRIGGER naam_trigger

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF naam_kolom]

ON naam_tabel

[REFERENCING OLD AS oud NEW AS nieuw]

[FOR EACH ROW]

WHEN (conditie)

BEGIN
```

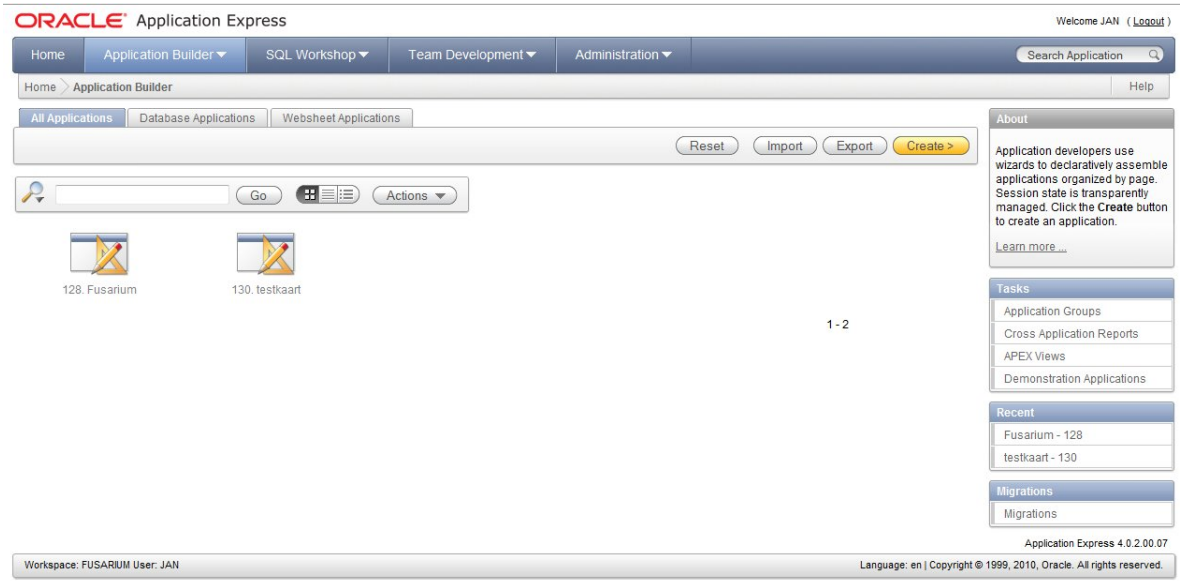
```
    --sql-statements

END;
```

De sleutelwoorden **BEFORE**, **AFTER** en **INSTEAD OF** duiden aan op welk moment de *trigger* moet uitgevoerd worden ten opzichte van de datamanipulatie. Het laatste sleutelwoord **INSTEAD OF** is het enige dat toegelaten is bij databankviews. Verder wordt gespecifieerd welke combinatie van datamanipulaties (**INSERT**, **UPDATE** of **DELETE**) de uitvoering van de *trigger* veroorzaken. Bij een *update trigger* kan ook de naam van een kolom meegegeven worden, waarbij deze procedure enkel uitgevoerd wordt als die specifieke kolom van een tabel wordt aangepast. De waarden in de tabel voor en na de verandering van de data zijn respectievelijk beschikbaar als de gebonden variabelen :**old**.naam_kolom en :**new**.naam_kolom. De referentie van de oude en nieuwe rij kunnen echter ook hernoemd worden. Er is logischerwijs geen referentie van de oude toestand van de rij beschikbaar wanneer een nieuwe rij wordt toegevoegd aan de tabel (**INSERT**). Vice versa is er geen referentie van de nieuwe toestand van de rij als deze verwijderd wordt (**DELETE**). Er kan ook nog onderscheid gemaakt worden tussen *row level triggers* en *statement level triggers*. De *row level trigger* wordt aangeduid met de clausule **FOR EACH ROW** en hierbij wordt de code in de PL/SQL-blok uitgevoerd per rij die gemanipuleerd wordt door één SQL-expressie. In een *statement level trigger* wordt de code éénmalig uitgevoerd. Bij de *row level trigger* kan ook een conditie gespecifieerd waaraan per rij moet voldaan zijn om de code uit te voeren.

In het Oracle-databanksysteem is het ook mogelijk om code geschreven in de programmeertaal Java te gebruiken in PL/SQL-code. Er kan in de OracleJVM (*Oracle Java Virtual Machine*) op het databanksysteem *Webservice*-client software opgestart worden die een *Webservice* op een externe host kan aanroepen. Men kan daarnaast ook rechtstreeks publieke statische methodes van Java-klassen in de OracleJVM rechtstreeks aanroepen van Java-clienten zonder gebruik te maken van de JDBC-API (*Java DataBase Connectivity*) of SQLJ (een ISO-standaard (ISO/IEC 9075-10) om SQL-uitdrukking in Java-programma's te integreren) (7). Hierbij wordt gebruikt gemaakt van het JPublisher-hulpmiddel voor de creatie van een client-proxy-klasse met dezelfde signatuur als de Java-klasse aan de zijde van de server. Eens een instantie van deze client-proxy-klasse met JDBC-verbinding wordt aangemaakt, kan men de proxy-methodes rechtstreeks aanroepen. Voor complexere berekeningen kan men gebruik maken van *Enterprise Java Beans* (EJB) die ingezet zijn op een externe server via Java-RMI (*Remote Method Invocation*). Dit is een Java-interface die het objectgeoriënteerd equivalent is van een *remote procedure call* (RPC). Voor de integratie van andere programmeertalen bestaat de mogelijkheid om gebruik te maken van de *Oracle Call Interface* (OCI) of uitbreidingen hierop (9).

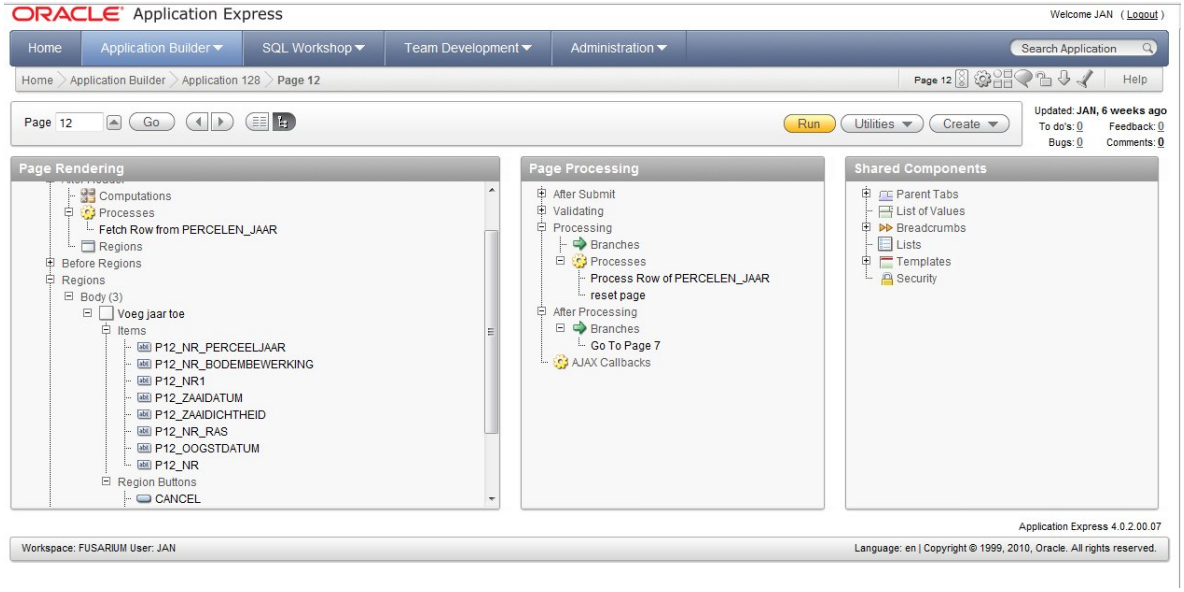
2.3 APEX en AnyChart

Oracle Application Express (afgekort als Oracle APEX) is een licentievrije webgebaseerde ontwikkelingsomgeving gebouwd rond het databanksysteem van Oracle (zie Figuur 2.3). De *serverside* scriptingtaal en het conditioneel tonen van bepaalde pagina's of pagina-elementen is gebaseerd op SQL-statements en PL/SQL-procedures. Deze procedures kunnen anonieme blokken zijn of *stored procedures*. Er zijn reeds veel basisfunctionaliteiten geïmplementeerd zoals authenticatie- en autorisatieschema's en navigatie-elementen – zoals tabbladen of *bread crumbs*.



Figuur 2.3: Na het inloggen tot de APEX-ontwikkelingsomgeving krijgt men een overzicht van de applicaties die in *workspace* aangemaakt zijn en kan de ontwikkelaar deze ook beheren. Daarnaast kan men ook gebruik maken van de *SQL-workshop* voor het beheer van databankobjecten in de schema's waartoe de ontwikkelaar toegang heeft gekregen binnen APEX.

Een applicatie is opgebouwd uit verschillende pagina's (zie Figuur 2.5. Aan deze pagina's kunnen verschillende buttons, page items, regio's, processen en *dynamic actions* toegevoegd worden. Buttons zijn de standaard html-elementen die kunnen gebruikt worden om formulieren op te bouwen, naar andere pagina's te navigeren en processen of *dynamic actions* te starten. Pagina-items zijn elementen die een bepaalde waarde bijhouden of waarin een waarde kan ingevuld worden (bijvoorbeeld voor een formulier). Daarnaast zijn er ook applicatie-items die over alle pagina's van de applicatie beschikbaar zijn (als het ware sessieobjecten). De waarden opgeslagen in deze items kunnen aangesproken op verschillende wijzen. De waarde van een pagina-item is in html-code of javascript (voor een *dynamic action*) beschikbaar als &naam_item. (dus beginnend met een ampersand en eindigend met een punt – dit laatste kan in teksten tot verwarring leiden) waar naam_item de naam van de pagina-item is. Hetzelfde is geldig voor applicatie-items, maar is dus over de gehele applicatie beschikbaar. SQL-statements en processen (PL/SQL-procedures of -functies) kunnen de waarde van de items opvragen op twee manieren meerbepaald via de SQL-functie V(naam_item) of als *bind variable* :naam_item (dus beginnend met een dubbelpunt) (13).



Figuur 2.4: Als men een individuele pagina bekijkt in de ontwikkelingsomgeving is te zien hoe deze opgebouwd is uit verschillende regio's, pagina-items en processen. De regio's zijn te vinden in de sectie *Page rendering* zoals bijvoorbeeld de regio "Voeg jaar toe" op deze figuur. Bij deze regio horen verschillende pagina-items zoals"P12_NR_PERCEELJAAR" en "P12_NR_BODEMBEWERKING". Op de figuur zijn ook verschillende processen te zijn zowel onder de sectie *Page rendering* (bijvoorbeeld "Fetch Row from PERCELEN_JAAR") als de sectie *Page Processing* ("Process Row of PERCELEN_JAAR")

Deze processen kunnen uitgevoerd worden op drie verschillende momenten namelijk voor het laden van de pagina, na het laden van de pagina en na het verlaten van de pagina. Voor het laden van de pagina kunnen berekeningen gedaan worden om bijvoorbeeld de waarde van een pagina-item te bepalen of kan bijvoorbeeld een *rowfetch* gedaan worden om een reeds ingevulde rij uit een databanktabel al in te vullen in een formulier voor de aanpassing van rij. Na het laden van pagina kunnen er asynchroon bepaalde gegevens geladen worden of gegevens verwerkt worden via een *on-demand-proces* dat aangesproken wordt via een *dynamic action* op deze pagina. Na het verlaten van de pagina (bijvoorbeeld door het indienen van een webformulier) kan er een proces uitgevoerd worden dat hoort bij een bepaalde knop op de oorsprongpagina en het verlaten van deze pagina veroorzaakte. Daarnaast kunnen er terug berekeningen gedaan worden of bepaalde items gereset worden. Als laatste wordt bepaald welke pagina vervolgens geladen wordt. Welke pagina na de *branch* geladen wordt, kan ook afhangen of bepaalde voorwaarden voldaan zijn.

De pagina's worden opgebouwd uit verschillende regio's die op zich dan weer kunnen bestaan sub-regio's (zie Figuur 2.4. Er zijn bepaalde basistypes van de regio's die via een wizard aangemaakt kunnen worden. Het kan hier ondermeer gaan om statische html-code, statische tabellen (*reports*) of dynamische tabellen (*interactive reports*), AnyChart-grafieken en -kaarten en de navigatie-elementen zoals *bread crumbs*. Verder zijn er ook regio's die toelaten om formulieren aan te maken om onder andere databanktabellen te manipuleren. Ze worden aangemaakt via een wizard waarbij om te beginnen een SELECT-uitdrukking moet gegeven worden met de te manipuleren tabel en zijn respectievelijk kolommen. De onwikkelaar moet ook aangeven welke operaties mogelijk zijn (UPDATE, INSERT en/of DELETE). Evenals moet aangegeven welke de primaire sleutels zijn van de tabel en hoe deze ingevuld moeten worden (een auto-increment-*trigger* die reeds is geïmplementeerd op de databank of een zelf te definiëren anonieme PL/SQL-blok). Een voorbeeld voor de tabel met de percelen in de webtool is

als volgt:

Broncode 2.7: Invulling van een primaire sleutel

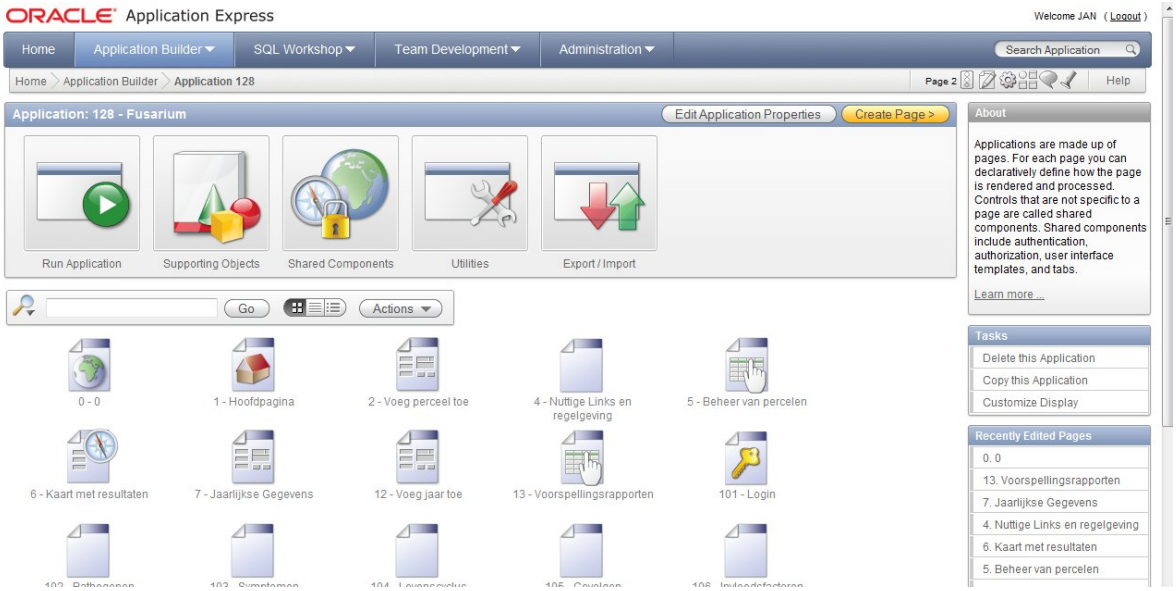
```
DECLARE
  FUNCTION get_pk RETURN varchar2
  IS
   L_PRIMARY_KEY NUMBER;
BEGIN
   SELECT MAX(NR) + 1
   INTO   L_PRIMARY_KEY
   FROM  WEB_PROD.WEB_PERCELEN;

   RETURN L_PRIMARY_KEY;
END;
BEGIN
  :P2_NR := get_pk;
END;
```

De pagina-items per kolom en processen die de aanpassingen verwerken, worden na het afwerken van de wizard toegevoegd aan de pagina. De pagina-items die de kolommen voorstellen, kunnen een gewoon invulveld zijn. Velden die niet rechtstreeks aangepast mogen worden door de gebruiker kunnen als verborgen item op de pagina te zetten. Voor velden die een datum voorstellen is het de beste optie om een datumkiezer van het veld te maken. Voor een veld dat niet vrij mag ingevuld worden door de gebruiker zoals categorische variabelen of vreemde sleutels kan er een selectielijst gedefinieerd worden. Het kan hier dan gaan om een statische lijst, maar men kan ook een dynamische lijst op basis van een *select*-uitdrukking gedefinieerd worden. Deze selectielijst bestaat uit de waarden die moeten getoond worden aan de gebruiker en de bijhorende waarden die in de tabel moeten ingevuld worden. Deze kolommen worden respectievelijk aangeduid met de namen *display* en *return* of afgekort *d* en *r* in de *select*-uitdrukking.

Broncode 2.8: Definitie van een selectielijst

```
SELECT
        naam d, NR r
FROM
        web_prod.web_teelten
ORDER BY 1
```

Regio's kunnen toegevoegd worden aan één enkele pagina (waar het tonen van de regio's afhangt van bepaalde condities) of op verschillende pagina's. Dit laatste is het geval als men een regio toevoegt aan de zogenaamde *page zero* van de applicatie. Standaard is de regio zichtbaar op elke pagina van de applicatie. Hieraan kunnen condities verbonden worden die de regio beperkt tot een door de applicatiebouwer te specifiëren lijst van pagina's. Deze lijst wordt gespecifieerd als een karakterreeks van paginanummers gescheiden door een puntkomma (zogenaamd *comma seperated values* of CSV).



Figuur 2.5: Als men een applicatie opent krijgt men overzicht van de pagina's die tot de pagina behoren en kunnen deze ook beheerd worden. Elke pagina heeft een unieke nummer die gebruikt wordt om deze te identificeren voor bijvoorbeeld *branches* na het verlaten van een pagina of het conditioneel tonen van bepaalde gedeelde regio's die op de *page zero* van de applicatie te vinden zijn.

AnyChart is een softwarepakket dat toelaat om dynamische kaarten, grafieken en dashboards aan webapplicaties toe te voegen. Het is langs de kant van de client gebaseerd op Adobe Flash player en meer recenter zijn er ook versies van de grafieken gebaseerd op de HTML5-canvas en javascript om platformonafhankelijke compatibiliteit te garanderen. De SWF-bestanden van AnyChart worden (mogelijk op een dynamische en asynchrone wijze) aangestuurd door een XML-document dat opgebouwd is volgens de documentdefinitie van AnyChart (men kan zich hierbij baseren op de documentatie (1) of de voorbeeldkaarten en -XML-bestanden van de AnyChartwebsite). Hierdoor kan in principe langs de kant van de server een XML-document gegenereerd worden volgens deze specificaties met om het even welke programmeertaal. Er zijn dan ook serverside scriptingtalen zoals PHP, ASP(.NET) en PL/SQL in geval van Oracle APEX waarmee AnyChart geïntegreerd is(2). Bij de integratie van AnyChart in APEX werd gekozen om dit terug via een wizard mogelijk te maken. Op basis van de keuzes van de ontwikkelaar wordt het SWF-document dan opgesteld. De datareeksen die op de kaarten/grafieken moeten verschijnen worden gegenereerd op basis van een door de ontwikkelaar op te geven SQL-query die een tabel oplevert waarvan de namen van de kolommen overeen komen met de namen van de datareeksen gedefinieerd in het XML-document. De datareeks die in XML-formaat wordt gegenereerd op basis van deze tabel is dan terug te vinden in het pagina-item data. De datareeks wordt dan toegevoegd aan het XML-document als &data. waardoor de inhoud van de respectievelijke pagina-item wordt ingevuld. Dit laat toe om de datareeks onafhankelijk van het XML-document te herladen. Dit en andere manipulaties gebeuren via javascript(10).

Hoofdstuk 3

Opbouw van de webtool

3.1 Analyse van de bestaande omgeving

In de volgende twee secties wordt de reeds aanwezige software en databanken van het *Fusarium*-project en de BDB besproken. Hiervan is ook een overzicht gegeven in een *Deployement*-diagram (zie Figuur 3.1) met ook een voorstel van hoe deze omgevingen verder zouden kunnen uitgebreid worden in functie van de ontwikkeling van deze applicatie.

3.1.1 Project *Fusarium*

Software en databanken

Databank van project *Fusarium*:

- Bereikbaar vanop UGent-netwerk of na het opzetten van VPN-verbinding met UGent-netwerk
- Databankversie: PostgreSQL 8.3.13
- phpPgAdmin 4.2.2 (PHP 5.2.6-1+lenny9)

Er zitten negen tabellen in het databankschema *headblight*:

- **Experimenten** In deze tabel staan alle resultaten van de rassenproeven 2002 tot nu van de Hogeschool Gent. De tabel bevat ook id's die verwijzen naar de tabellen die de omstandigheden van de proef beschrijven. Deze verschillende variabelen in deze tabel zijn.
  - Het DON-gehalte van de geoogste granen (ppm).
  - De *Fusarium* species die werden aangetroffen (-). Het gaat hier om vijf *dummy*-variabelen voor de species *Fusarium avenaceum*, *F. culmorum*, *F. graminearum*, *F. poae* en *Microdochium nivale*.
  - Een procentuele verdeling voor vijf ziekteklassen (%). Dit is een maat voor de aantasting van het perceel door *Fusarium*. De percentages in de vijf ziekteklassen werden bepaald door at random 100 aren per perceel te evalueren voor *Fusarium* symptomen. De indeling van een aar in één van de vijf ordinale ziekteklasses gebeurde op basis van een scoring van de oppervlakte van de aar die bedekt was met *Fusarium* symptomen (zie Tabel 3.1).
  - De id's voor de tabellen die rechtstreeks verbonden zijn met deze tabel (**Gewassen**, **Voorvrucht** en **Zaaijaar**).

– Het nummer van de herhaling als er verschillende herhalingen waren voor een ras.

Tabel 3.1: *Fusarium*-symptomen en de overeenkomstige ziekteklassen (gegevens van Hogeschool Gent)

Symptoom	Ziekteklasse (-)
Gezond	1
Tot 25% van de aar bedekt	2
25 tot 50% van de aar bedekt	3
50 tot 75% van de aar bedekt	4
75 tot 100% van de aar bedekt	5

- **Gewassen** In deze tabel staan alle rassen die in de loop van de jaren in de proeven zijn gebruikt. Deze proeven dienen ook ter evaluatie van deze rassen.
- **Voorvrucht** Per voorvrucht werd hierin een ordinale risicoklasse toegekend door de verantwoordelijken van de Hogeschool Gent om het risico uit te drukken dat er op deze vrucht en zijn oogstresten *Fusarium* zou kunnen ontwikkelen en overleven.
- **Zaaijaar** Per jaar en ook per proefveldlocatie zijn er verschillende omstandigheden die kunnen veranderen voor de proefvelden en deze worden dan ook aangeduid in deze tabel of gelinkt naar andere tabellen met deze gegevens:
  - de bodemcode die of het weerstation dat gebruikt wordt voor een zaailocatie kunnen bijvoorbeeld over de jaren heen veranderen en hiervoor wordt er dan ook gelinkt naar hun respectievelijke tabellen (**Bodemcodes**, **Weerstations** en **Zaailocaties**),
  - de zaaidichtheid waarmee het proefveld werd gezaaid,
  - de bodembewerking die op het proefveld werden toegepast na de voorvrucht,
  - het dagnummer van de oogst- en zaaidatum van die proefveldlocatie.
- **Bodemcodes** In deze tabel staat het percentage zand, leem en klei zoals ze werden afgelezen in de textuurdriehoek voor de gegeven bodemcode.
- **Zaailocaties** Per zaailocatie wordt de volledige naam vermeld. De tabel **Weermetingen_math** is hieraan rechtstreeks gelinkt.
- **Weerstations** In deze tabel staat de naam van de locatie van elke weerstation
- **Weermetingen** De weermetingen kunnen aan een zaailocatie per jaar gelinkt worden via de weerstation-id in de tabel **Zaaijaar**. Dit zijn de dagelijkse weergegevens van de weerstations van het netwerk van weerstations in Vlaanderen die door het Proefcentrum Aardappelenteelt (PCA) gebruikt worden in voorspellingen voor de aardappelziekte (*Phytopthora infestans*). Individuele instrumenten leggen deze gegevens per uur vast. Hiervan berekende de verantwoordelijke van het PCA dan dagelijkse gegevens. Het gaat hier over:
  - de gemiddelde, maximum- en minimumtemperatuur (°C),
  - de relatieve vochtigheid (%),
  - de neerslagsom (mm),
  - het aantal bladnaturen (uren),
  - het dagnummer van de meetdag (-),
  - het jaar (-).

Deze tabel heeft ook een id die verwijst naar de **Weerstations**-tabel.

- **Weermetingen_math** Voor elke zaailocatie zochten de verantwoordelijken van Hogeschool Gent in het programma Mathematica per jaar weerstations die zo dicht mogelijk bij de zaailocatie lagen en zoveel mogelijk complete weergegevens hadden. Hiervan werden dan de dagelijkse weergegevens berekend. Deze gegevens dienen om onvolledige tijdreeksen van de tabel **Weermetingen** aan te kunnen vullen. Deze tabel bevat de volgende variabelen:
  - de gemiddelde, maximum- en minimumtemperatuur (°C),
  - de gemiddelde, maximum en minimum relatieve vochtigheid (%),
  - de neerslagsom (mm),
  - de gemiddelde, maximum- en minimumdauwpunttemperatuur (°C) (deze variabele is een maat voor de hoeveelheid vocht in de lucht),
  - de gemiddelde, maximum- en minimumwindsnelheid (km/h),
  - de gemiddelde, maximum- en minimumluchtdruk (barometrische druk uitgedrukt in millibars),
  - het dagnummer van de meetdag (-),
  - het jaartal (-).

  Deze tabel is gelinkt aan de **Zaailocaties**-tabel.

3.1.2 Bodemkundige Dienst van België

Software en databanken

Databank van de Bodemkundige Dienst van België:

- Bereikbaar vanop BDB intern netwerk of na het opzetten van VPN-verbinding met BDB-netwerk
- Databankversie: Oracle Database 10g
- Oracle Aplication Express 4.0.2.00.07

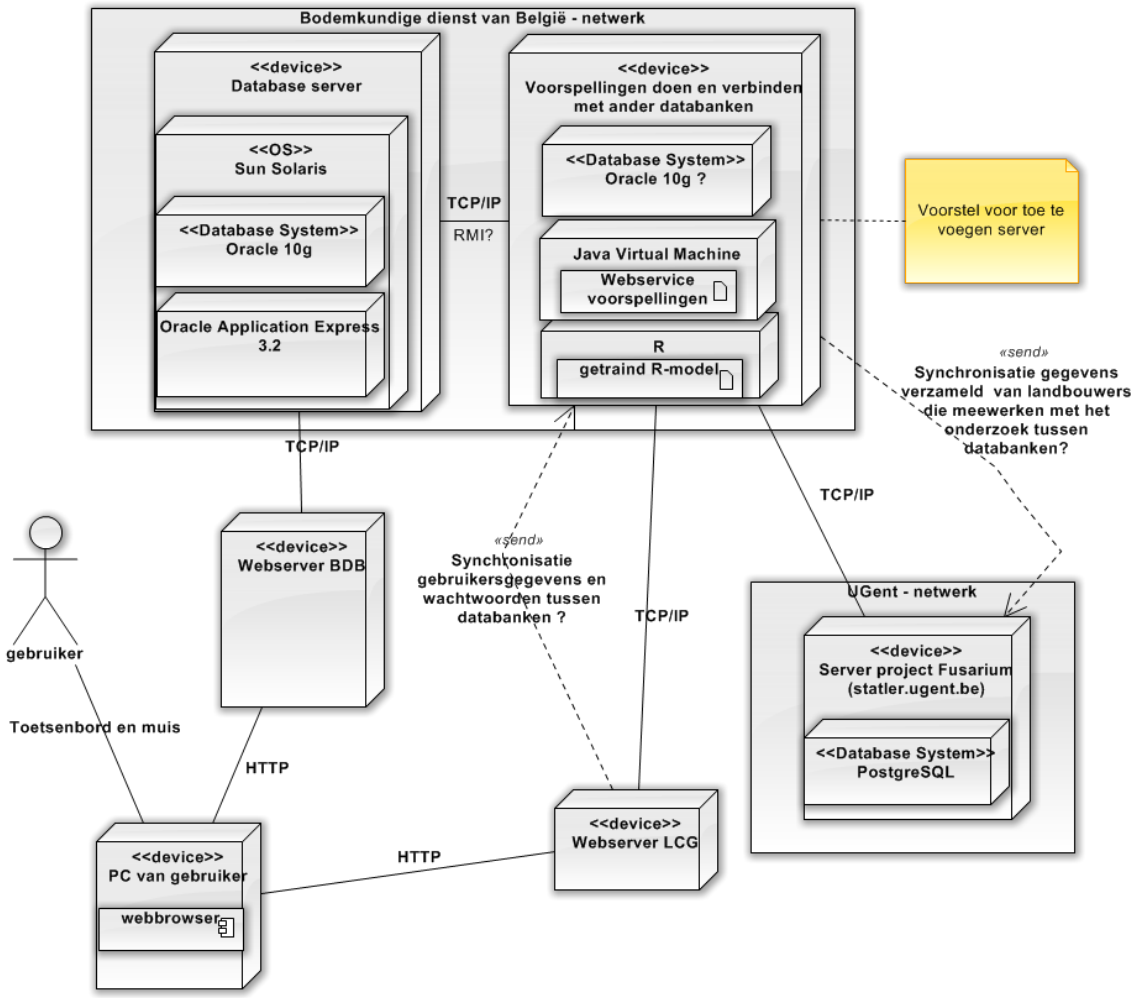
Op deze databank zijn er drie schema's die reeds relevante tabellen bevatten in verband met de webtool:

- *web_prod*: Deze bevat de tabellen gerelateerd aan de andere webapplicaties van de BDB.
  - **web_bedrijven**: (landbouw-)bedrijven die gebruik maken van de webapplicaties: bevat naam, adres (straat/huisnummer/busnummer) en een referentie naar **prt_gemeenten** (gem_nr).
  - **web_bedrijven_contacten**: tabel om de 1..*-1..*-relatie tussen **web_bedrijven** en **prt_contacten** voor te stellen. Meerdere (familie-)leden kunnen bij eenzelfde bedrijf horen en één gebruiker kan tot meerdere bedrijven behoren.
  - **web_rechten**: tabel om de 1..*-1..*-relatie tussen **web_bedrijven** en **prt_contacten** voor te stellen waar de contacten gewasadviseurs (of gelijkaardig) zijn. Deze krijgen via deze tabel rechten om in de plaats van de landbouwer zijn bedrijf te beheren.
  - **web_login**: de login- en andere gebruikersgegevens (taal, email-adres) van de gebruikers van de webapplicatie die gebruikt worden voor het inloggen. Het wachtwoord wordt in plain text opgeslagen.
  - **web_percelen**: de individuele percelen die bij een bedrijf horen. De landbouwer geeft hieraan een naam. Daarnaast voegt de landbouwer informatie toe zoals de straat en relatieve ligging (bvb. achter de stal) . Verder wordt er zoals in **web_bedrijven** gerefereerd

naar de tabel **prt_gemeenten**. Er wordt ook verwezen naar de tabel **web_grondsoorten** via **web_grondsoorten_cp**.
  - **web_grondsoorten**: deze tabel bevat de verschillende grondsoorten. Deze tabel wordt gebruikt voor de applicatie in verband met het mestactieplan (MAP).
  - **web_teelten**: deze tabel bevat de verschillende teelten (dus ook de verschillende granen en voorvruchtgewassen). Er is ook een verwijzing naar een tekstbestand met de gewascategoriën waartoe dit gewas behoort.

- *prtdev*: Informatie over gemeenten en over gebruikers (voor onder andere facturatie).
  - **prt_contacten**: tabel met personen die gebruik maken van de diensten van de BDB.
  - **prt_gemeenten**: de verschillende gemeenten. Bij de Belgische gemeentes staat ook de postcode en NIS-code. Daarnaast kan deze tabel informatie bevatten over de provincie en het gewest (Vlaams/Frans/Brussels). Deze tabel bevat 16456 gemeenten, waarvan de 2860 Belgische gemeentes met hun NIS-code, provincie en gewest gegeven zijn. Deze tabel verwijst ook naar zichzelf (deelgemeenten naar hoofdgemeente) en de tabel **prt_landen**.
  - **prt_gemeenten_t**: bevat de Franstalige naam van de (tweetalige) gemeenten uit het Brussels gewest in de tabel **prt_gemeenten**.
  - **prt_landen**: tabel met de verschillende landen en hun officiële ISO-code.
  - **prt_landen_t**: bevat niet-Nederlandse naam van een land uit de tabel **prt_landen**.

- *exp_inout*: Weergegevens, weervoorspellingen en informatie over de weerstations die de BDB aankoopt bij het Koninklijk Meteorologisch Instituut van België (KMI).
  - **meteo_stations_def**: definitie van de weerstations waarvan de weermetingen komen. Hierbij zijn er ook gegevens over de geografische ligging van het weerstation (hoogte, lengte- en breedteligging).
  - **meteo_ecmwf_reg**: dagelijkse weersvoorspellingen van KMI. Er wordt verwezen naar het weerstation uit **meteo_stations_def** met volg_nr. Deze tabel gaat terug tot het 2008.
  - **meteo_ecmwfcor_reg**: gegevens uit **meteo_ecmwf_reg** aangevuld met extra gegevens over de oorzaak van de neerslag.
  - **penman_db**: dagelijkse weermetingen per weerstation gecombineerd met geïnterpoleerde weergegevens. Het gaat hier om metingen van in het 1993 tot het heden.
  - **meteo_metagri**: dagelijkse temperatuurmetingen per weerstation die ondermeer de basis vormen voor de waarden in **penman_db**.
  - **meteo_uurwaarden**: uurlijkse weermetingen per weerstation die evenals **meteo_metagri** de basis vormt voor **penman_db**.



Figuur 3.1: Overzicht van alle hardware en software in een *deployement* diagram

3.2 Functioneel ontwerp

De verschillende functionaliteiten die gewenst zijn in de webtool zijn in de volgende secties gegroepeerd en uitgeschreven waarbij er ook een overzicht wordt gegeven in Figuur 3.2.

3.2.1 Inloggen op de webtool

Op de website van de Bodemkundige Dienst van België (BDB) kunnen nieuwe gebruikers een BDBnet-account aanmaken en inloggen op de webtool. Bestaande BDBnet-gebruikers kunnen hun bestaande account gebruiken. De gebruikers van het Landbouwcentrum Granen (LCG) kunnen op deze manier ook een account aanmaken om de webtool te gebruiken waarbij mogelijk een verantwoordelijke van LCG deze accounts kan beheren of toestemming geven tot bepaalde applicaties. De gebruikers van de website kunnen pagina's bekijken met:

- algemene informatie over de *Fusarium*-problematiek,

- de onderzoeksresultaten van de proeven van de Hogeschool Gent in verband met *Fusarium*, deze worden ondermeer voorgesteld door bladerbare kaarten,
- nuttige links over bijvoorbeeld de regelgeving.

Een gebruiker kan zijn:

- een landbouwer die de percelen van zijn eigen bedrijf beheert,
- familieleden die (voor het verkrijgen van bepaalde subsidies) de percelen van de landbouwer beheren,
- een gewasadviseur die de percelen van verschillende landbouwers beheert om hen te adviseren,
- een graanhandelaar die gerichte staalnames wil doen om DON-gehaltes in het graan te bepalen.

Een gebruiker heeft:

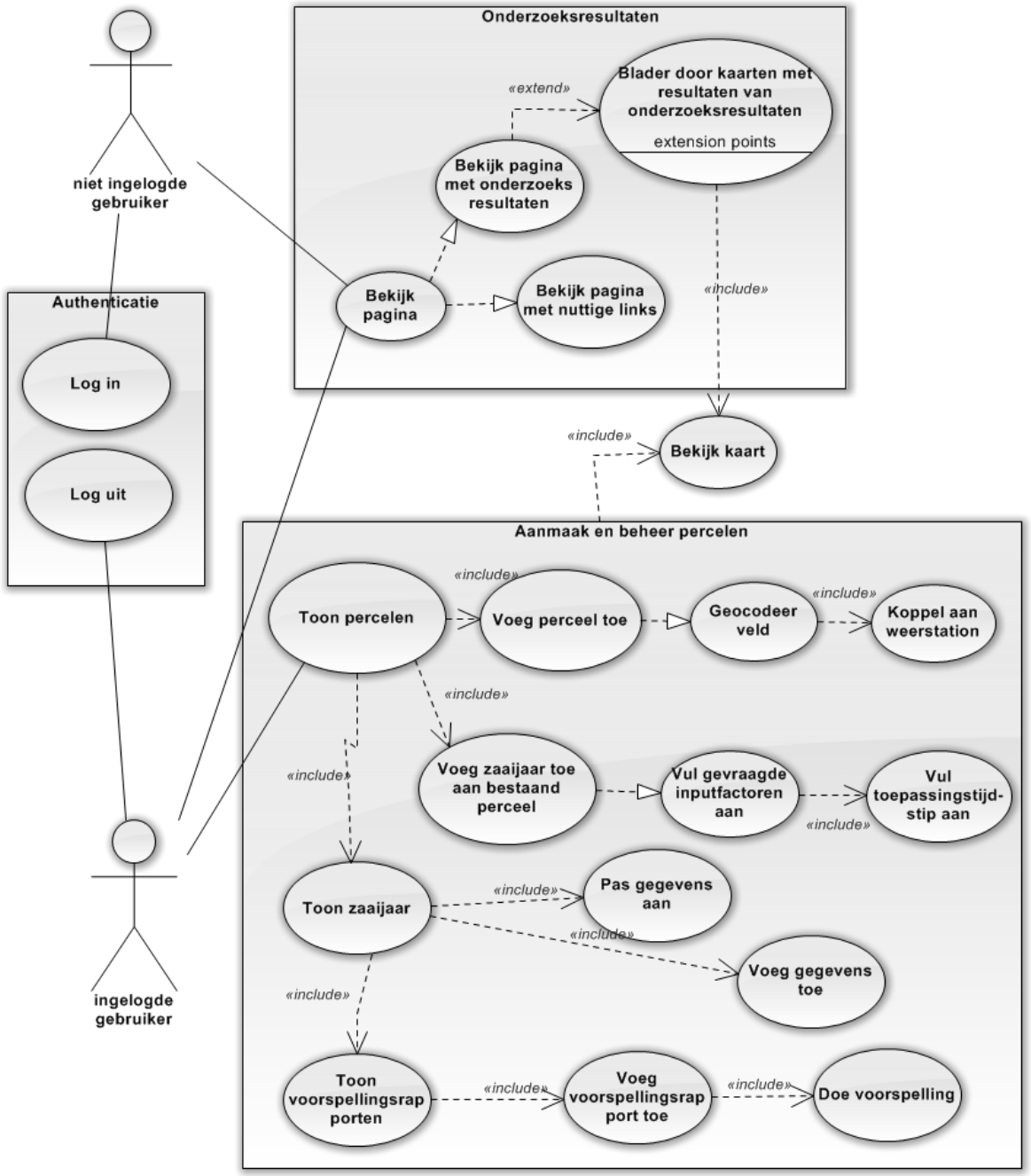
- een voornaam en familienaam,
- een email-adres,
- een wachtwoord.

3.2.2 Persoonlijk profiel

De landbouwer kan nieuwe percelen toevoegen aan zijn profiel/bedrijf. Een perceel heeft een paar vaste eigenschappen bepaald door het adres van het perceel. Hiervoor moet de straat, huisnummer en gemeente opgegeven worden. Daarbij wordt het perceel ook een benaming gegeven zoals bijvoorbeeld "achter de stal". Bij een perceel hoort een weerstation (het dichtstbijzijnde) waarvan de weergegevens gebruikt worden als input voor de voorspellingsmodellen. Dit weerstation wordt gekozen door de geografische coördinaten van de gemeente waarin het perceel ligt te vergelijken met deze van het weerstation. Bij de keuze van het weerstation kan ook rekening worden gehouden met de geografische streek (bijvoorbeeld bij de kuststreek, waar de zee een invloed op het weer heeft) of de hoogte ten opzichte van het zeeniveau. Daarnaast kan voor de weergegevens het gewogen gemiddelde genomen worden van de drie dichtstbijzijnde weerstations.

Daarnaast wordt elke Belgische gemeente geïdentificeerd door een NIS-code. Uit deze alfanumerieke code kan op basis van het eerste getal de provincie afgeleid worden en in combinatie met het tweede getal het arrondisement. De laatste drie cijfers en letter identificeren elke deelgemeente op een unieke wijze.

Per jaar (het jaar waarin het tarwegewas wordt geoogst) zijn ook verschillende factoren die de ontwikkeling van *Fusarium* kunnen beïnvloeden zoals voorvrucht, tarweras en fungicidegebruik. Daarbij horen ook de (geplande) tijdstippen van toediening van fungicide, zaaien of oogsten. Totdat de gebruiker deze handelingen fysiek uitgevoerd heeft (dus het geplande tijdstip is voorbijgegaan), kan de landbouwer deze waarden aanpassen. Het ras en de voorvrucht zijn voorbeelden van een categorische variabele, hieraan kan ook een risicoklasse voor de ziekte verbonden worden. De landbouwer kan op deze manier elk jaar aan de reeds gecreëerde percelen nieuwe gegevens van het groeiseizoen (voorvrucht, fungicidebehandeling, ...) toevoegen.



Figuur 3.2: Overzicht van alle hardware en software in een *deployement* diagram

3.2.3 Voorspelling

Op basis van de informatie geleverd door de gebruiker kunnen verschillende voorspellingen/simulaties van de aantasting door *Fusarium* gedaan worden. Er kan onderscheid worden gemaakt tussen:

- voorspelling in de winter. Dit gebeurt op basis van de weergegevens die waargenomen werden tot op het moment van het opvragen van het rapport en historische weergegevens voor de periode

erna: deze historische weergegevens zijn enerzijds van een jaar met hoge DON-gehaltes en een jaar met lage DON-gehaltes, er kan ook nog een onderscheid worden gemaakt tussen het niet en wel behandelen met fungiciden,
- een tweede voorspelling rond de periode van de bloei van het tarwegewas. Hierbij wordt naast de historische weergegevens ook rekening gehouden met de weersvoorspellingen rond deze periode. De voorspelling gebeurt enerzijds op basis van het door de modelbouwer geleverde meest recente model. Anderzijds moeten dezelfde voorverwerkte variabelen gebruikt worden als bij de training van het respectievelijke model. Het gaat hier dus zowel om de factoren geleverd door de landbouwer als de voorverwerkte tijdreeksen van de weergegevens die bij het perceel horen.

De factoren (of mogelijk ook continue variabelen, data van handelingen) die gevraagd worden aan de landbouwers moeten dus ook ingesteld worden door de modelbouwer. De voorverwerking van deze variabelen kan zijn:

- maandelijkse samenvattingen van dagelijkse weergegevens: gemiddelde (genormaliseerde som), percentielen, Tuckey Five Number-statistieken of aantal dagen waarop een bepaalde limietwaarde overschreden werd,
- dagnummer van een datum,
- omzetting van factorvariabele naar:
  - de ordinale risicoklasse
  - codering als verschillende dummyvariabelen (waarde nul of één).

Het model kan ook verschillende types of combinaties van outputvariabelen hebben (te specifiëren door de modelbouwer) en kan ook gemodelleerd zijn met verschillende programmeertalen.

Er wordt een rapport gegenereerd dat één of meer voorspellingen bevat. Afhankelijk van de snelheid van het voorspellingsmodel of de voorkeuren van de gebruiker of modelbouwer wordt:

- Een dynamische HTML-pagina gegenereerd als rapport.
- Een e-mail gestuurd naar de landbouwer zodra de (nieuwste) voorspelling aanwezig is, het overschrijden van een bepaald maximumniveau of eender welke andere trigger. Dit kan zijn als een HTML-gebaseerde email of een pdf-document.

In dit rapport wordt informatie gegeven over hoe de waarde van voorspelling zich verhoudt tot bepaalde maximumniveau's en wordt er ook mogelijk al advies gegeven over hoe mogelijk verder te handelen.

De landbouwer kan ook opteren om mee te werken aan het onderzoek. Hij kan dan een eigen waarneming doen op een van zijn eigen percelen. Het kan hier gaan om verschillende variabelen zoals de ziekte-index van de besmetting met *Fusarium* of het tijdstip van de bloei. De landbouwer kan zijn eigen waarnemingen dan vergelijken met de (geanonimiseerde) waarnemingen van andere landbouwers (die onder gelijkaardige omstandigheden werken), de voorspellingen of de voorgeschiedenis van zijn respectievelijke perceel. De ingevulde gegevens van de eigen waarnemingen worden vergeleken met (voorverwerkte statistieken van) data reeds aanwezig in de databank. Als het hier om een *outlier* of gedupliceerde waarneming zou gaan, kan dan mogelijk ook gevraagd worden om een nieuwe waarneming te doen of de gegevens te corrigeren.

3.3 Technisch ontwerp

3.3.1 Logisch databankontwerp

Voor het logische ontwerp van de databank bij de webtool werd getracht zoveel mogelijk de bestaande tabellen van de Bodemkundige Dienst en het project van Fusarium te hergebruiken. Hiervoor werd de datadefinitiescript van beide databanken geëxporteerd uit de databanksystemen. Voor de uitbouw van het logische en relationeel datamodel werd Oracle SQL *Developer Data Modeler* geïnstalleerd aangezien alle tabellen in de Oracle-databanksysteem van de BDB geïmplementeerd moeten worden en deze software al de basis van een datadefinitiescript voor de nieuwe tabellen kan genereren. De DDL-scripts werden geïmporteerd als relationeel model in de software (zie Figuren B.2 en B.4). Dit relationeel model kan dan omgezet worden naar het logische databankmodel (zogenaamd "reverse engineering") (zie Figuren B.1 en B.3). Deze laatste omzetting is niet volledig aangezien de tabellen die de *..*-relaties voorstellen niet omgezet wordt tot een relatie in het logische model. Aan dit logisch model kunnen via de grafische omgeving van de *Data Modeler* dan nieuwe entiteiten en relaties toegevoegd worden (zie Figuur B.5).

3.3.2 Relationeel databankontwerp

Na de uitbreiding van het logische ontwerp van de databank kan hieruit het relationeel model gegenereerd worden (zie Figuur B.6). Op basis van dit relationeel model kan het DDL-script, aangepast voor de Oracle-databanken, gegenereerd worden. Het ontwerp van het relationeel en logisch model wordt ook opgeslagen als een Oracle SQL Developer Data Modeler Design (.dmd). Hierdoor kunnen beide terug aangepast worden of later omgezet worden naar een formaat dat importeerbaar is in andere modeleersoftware voor de generatie van DDL.

3.4 Implementatie

3.4.1 Databanktabellen en *views*

Alle toegevoegde tabellen in verband met de webtool bevinden zich op de databanken van de BDB in het schema *fusarium*. Enerzijds zijn er de tabellen van het project *Fusarium* met de gegevens van de experimenten die aangepast zijn aan het Oracle-databanksysteem en waarvan de gegevens ook zijn gemigreerd. Het grootste aanpassing is dat het datatype boolean niet in het Oracle-databanksysteem gebruikt wordt. Hierdoor moesten de kolommen in de tabel *Experimenten* die de aanwezigheid van de verschillende *Fusarium* species voorstellen omgezet worden van het datatype boolean naar een karakterreeks van lengte één waarbij de waar en vals vervangen werden door respectievelijk 't' en 'f'. Verder moest de auto-increment-*trigger* en -*sequence* van de primaire sleutels van verschillende tabellen manueel aangemaakt worden. Daarnaast zijn er volgende tabellen toegevoegd:

- **percelen_jaar**: de gegevens van één groeiseizoen van wintertarwe voor één bepaald perceel. Een groeiseizoen hoort bij één bepaald jaar. Dit zijn gegevens die nodig zijn om elk jaar opnieuw een voorspelling te kunnen doen. In deze tabel komen ook de bijkomende observaties die de landbouwer kan doen om mee te werken aan het onderzoek. Voor het huidige voorspellingsmodel is het nodig dat de volgende gegevens worden geregistreerd:
  - zaai- en oogstdatum van het groeiseizoen,
  - zaaidichtheid

  - het tarweras (met een verwijzing naar de gewassen-tabel oorspronkelijk van het project *Fusarium*),
  - het voorvrucht (met een verwijzing naar de *web_prod*.**web_teelten** tabel. Deze bevat een veel uitgebreidere lijst met mogelijke voorvruchten dan de originele voorvrucht-tabel van het project *Fusarium*.
  - Het perceel waar dit groeiseizoen bijhoort. Dit zorgt dan ook voor de link naar weergegevens.

  Daarnaast kunnen de volgende gegevens bijkomend geregistreerd worden.

  - observatie van de bloeidatum
  - fungicidebehandeling rond het tijdstip van de bloeidatum en tijdstip van deze behandeling
  - de ziekte-index als de verschillende ziekteklassen

- **fungiciden**: tabel met de fungiciden en een mogelijk toe te voegen risicoklasse.
- **bodembewerking**: tabel met de verschillende soorten bodembewerkingen en een mogelijk toe te voegen risicoklasse.
- **voorspellingsrapport** Deze tabel bevat alle voorspellingen die op eenzelfde moment gedaan zijn en hoort bij de weers-, groeiseizoen- en perceelgegevens die op dat moment in de databank zijn ingevuld. Er wordt dan ook verwezen naar de tabel **percelen_jaar**.
- **voorspellingen** Voorspellingen die bij een voorspellingsrapport horen bevinden zich in deze tabel. Deze voorspelling stelt een outputvariabele van een bepaald voorspellingsmodel voor, en bevat dus ook een verwijzing naar de tabel met modellen. Bij de voorspelling kan er ook invulling gebeuren met historische weersgegevens of een verschillende waarden van een inputvariabele (zoals de oogstdatum of fungicidebehandelingen) gesimuleerd worden. In geval de invulling van de weersgegevens zit er een vlag in de tabel die aanduidt of het om de invulling van de weersgegevens van een slecht of een goed jaar gaat.
- **model** De voorspellingen zijn de outputvaribele van één bepaald type model getraind op een bepaalde dataset. Deze dataset kan veranderen over de tijd en een model kan daarop hertraind worden. Aangezien een model kan veranderen over de tijd wordt de datum van het toevoegen aan het databanksysteem bijgehouden, evenals om welke outputvariabele het gaat. Daarbij wordt informatie bijgehouden over de bestand(en) bij het getraind model horen, die voor het bekomen van een nieuwe voorspelling mogelijk zijn.
- **parameters** Voor eenvoudige lineaire modellen kan in deze tabel per inputvariabele de parameters opgeslagen worden. Het bevat de waarde van de parameter en tot welk inputvariabele en model deze parameter behoort.
- **model_weblogin** De rechten voor het toevoegen van een nieuw getraind model aan de tabel **model** wordt per outputvariabele vastgelegd in deze tabel.
- **weerstations_gemeenten** Per gemeente in de tabel *prtdev*.**prt_gemeenten** zijn de drie dichtstbijzijnde weerstations uit *exp_inout*.**meteo_stations_def** toegevoegd.
- **gemeenten** Een kopie van *prtdev*.**prt_gemeenten**.
- **gemeenten_grp** De arrondissementen van belgië en hun NIS-code.

Verder is er ook een *global temporary* tabel die gebruikt wordt in de procedures voor het bepalen van de dichtstbijzijnde weerstations bij een bepaalde locatie. Hierin kunnen gegevens tijdelijk in opgeslagen en gemanipuleerd worden net zoals een gewone databanktabel, met het verschil dat alle rijen uit deze tabel verwijderd worden na het wegschrijven van de resultaten uit het werkgeheugen (na een *commit*). Deze tabel wordt met de volgende syntax aangemaakt:

Broncode 3.1: tijdelijk_ws_gem

```
CREATE GLOBAL TEMPORARY TABLE FUSARIUM.TIJDELIJKE_WS_GEM
(
  GEM_NR NUMBER(10),
  METEO  NUMBER(5),
  GEW    NUMBER
)
ON COMMIT DELETE ROWS
NOCACHE;
```

Er zijn uiteindelijk ook verschillende *views* op de tabellen aan het databanksysteem toegevoegd. Het gaat hier SELECT-uitdrukkingen die frequent opgevraagd zullen worden of die moeten aangepast kunnen worden zonder de *triggers* of *procedures* die er van afhangen te moeten aanpassen. Voor de voorstelling van de resultaten uit de tabel **Experimenten** per arrondissement in een AnyChart-kaart werden twee *views* aan de databank toegevoegd:

- **Samenvattingexperimenten** (broncode C.1) Berekent de ziekte-index en de aanwezigheid van DON-producerende *Fusarium* species op basis van metingen in de tabel **Experimenten** en bevat ook het gemeten DON-gehalte. Voor de koppeling naar de regio's wordt ook de referentie naar het zaaijaar meegenomen in de *view*.
- **kaart** (broncode C.2) Koppelt de experimenten uit **samenvattingexperimenten** via de tabellen **zaaijaar** en **zaailocatie** aan de tabel **gemeenten**. Via de gemeentes kunnen de zaailocaties gekoppeld worden aan arrondissementen in **gemeenten_grp** en de gemiddelden van de metingen per arrondissement berekend worden. De naam van de arrondissementen worden dan gebruikt om op de AnyChart-kaart de gemiddelden bij de regio's op de kaart weer te geven.

Voor de koppeling van de weergegevens aan de gegevens van de experimenten (via de tabel **weerstations_gemeenten** die de gemeente van een perceel linkt aan de dichtstbijzijnde weerstations) zijn er ook twee *views*:

- **beschikbare_weerstations** (broncode C.3) Voor de invulling van **weerstations_gemeenten** mogen alleen de weerstations uit *exp_inout*.**meteo_stations_def** gebruikt worden waarvan er weermetingen in *exp_inout*.**penman_db** zitten voor de jaren die relevant zijn voor de modellen (de data van de experimenten gaat maar terug tot 2001). De lengte- en breedtegraad wordt ook omgerekend naar dezelfde eenheid als deze van de tabel **gemeenten**.
- **maandelijksewaarden** (broncode C.3) Per jaar en weerstation worden de maandelijkse statistieken van de weersgegevens uit *exp_inout*.**penman_db** berekend.

Bij het opvragen en opvolgen van voorspellingsrapporten horen drie *views*:

- **beschikbaremodellen** (broncode C.5) Bij het opvragen van voorspellingen mogen enkel de meest recente modellen per uitkomstvariabele uit de tabel **model** selecteerbaar zijn voor de gebruiker.
- **status_rapporten** (broncode C.6) Aangezien de berekeningen voor verschillende modellen sterk in uitvoeringstijd zou kunnen verschillen, wordt het percentage afgewerkte voorspellingen per voorspellingsrapport berekend.
- **rapporten_vandaag** (broncode C.7) Via een regio die over de gehele applicatie zichtbaar is (deze wordt toegevoegd aan de zogenaamde *page zero*) kan de gebruiker de status opvolgen van de rapporten die hij diezelfde dag heeft opgevraagd.

3.4.2 Databanklogica

Bij de tabellen die oorspronkelijk in de databank van het project *Fusarium* zaten, waren er voor de grootste tabellen (zoals **zaaijaar** en **experimenten**) auto-increment-*triggers* voor hun primaire sleutels. Deze werden in het Oracle-databank systeem gereproduceerd door een *sequence*-databankobject en een *trigger* te definiëren per tabel. Een *sequence* is een databankobject dat een opeenvolgende reeks getallen oplevert. Met het sleutelwoord *currval* kan de huidige waarde van het *sequence*-object opgevraagd worden en met het sleutelwoord *nextval* wordt de waarde in het *sequence*-object met een voorgedefinieerde aantal verhoogd en deze waarde opgevraagd. Dit verzekert dat als verschillende gebruikers op hetzelfde moment gegevens aan een tabel willen toevoegen ze elk een verschillende waarde krijgen voor de primaire sleutel.

Voor het toevoegen van een nieuw groeiseizoen mag de gebruiker dat maar éénmaal per jaar en per perceel doen en enkel voor het huidige jaar. Dit om te verhinderen dat de gebruiker verschillende redundante groeiseizoenen aan één perceel toevoegt of een groeiseizoen voor een verkeerd jaar definieert (en dus verkeerde weersgegevens gebruikt voor de voorspellingen). Er is dan ook een *unique constraint* op de tabel **percelen_jaar** die verzekert dat er enkel unieke combinaties van jaar en de vreemde sleutel naar de tabel *web_prod*.**web_percelen** kunnen toegevoegd worden. Verder is er een *trigger* (zie broncode C.8) die automatisch de primaire sleutel en het jaar invult bij het toevoegen van een nieuw *record* aan de tabel. Het jaar dat ingevuld wordt is afhankelijk van de maand dat de gebruiker dit groeiseizoen toevoegt. Als de maand na het einde van het vorige groeiseizoen (na augustus) wordt aangemaakt, wordt het volgende jaar ingevuld. Als dit gebeurt voor het einde van het groeiseizoen (voor of tijdens augustus) wordt het huidige jaar ingevuld. Dit laat toe dat de gebruiker groeiseizoengegevens toevoegt na het zaaien of bij de planning hiervan in het jaar voor het groeiseizoen en ook al voorspellingen opvraagt en verhindert dat hij meer dan een maand na het oogsten nog het groeiseizoen van deze oogst kan toevoegen of voorspellingen opvragen voor een afgelopen groeiseizoen.

Voor het opvragen van van een voorspellingsrapport wordt het meest recente model voor een uitkomstvariabele gekozen uit **model** via de *view* **beschikbaremodellen** (broncode C.5). In de tabel **model** wordt de creatiedatum automatisch toegevoegd door een *trigger* (broncode C.9). De pagina van de applicatie die instaat voor het toevoegen van voorspellingsrapporten bevat een formulier-regio die instaat voor het toevoegen van het voorspellingsrapport en de voorspellingen aan de databank. De pagina-items van dit formulier worden niet automatisch verwerkt, maar door een zelfgedefinieerde *stored procedure* (broncode C.14). In deze procedure wordt een voorspellingsrapport toegevoegd aan de tabel **voorspellingsrapport** waarbij het tijdstip wordt ingevuld waarop dit rapport is opgevraagd door een *trigger* (broncode C.10). Dit voorspellingrapport wordt ook gelinkt aan een groeiseizoen uit **percelen_jaar** via een vreemde sleutel die opgegeven werd als parameter van de procedure. Daarna worden in de tabel **voorspellingen** twee scenario's (een goed en een slecht jaar) voor de voorspelling toegevoegd voor het model dat werd meegegeven als parameter van de procedure. Hierbij wordt de primaire sleutel van het zojuist gecreëerde voorspellingsrapport toegevoegd aan de voorspellingen als vreemde sleutel. Het is goed denkbaar dat in plaats van één modelnummer voorspellingen worden toegevoegd voor meerdere uitkomstvariabelen in één voorspellingsrapport via een meervoudig selecteerbare lijst voor de modellen. Bij het toevoegen van een voorspelling is er een *trigger* (broncode C.11) die de een willekeurige waarde voor de voorspelling invult, maar die uiteindelijk zou moeten vervangen worden door het opvragen van een procedure die de voorspelling voor het gespecificeerde model en groeiseizoen genereert.

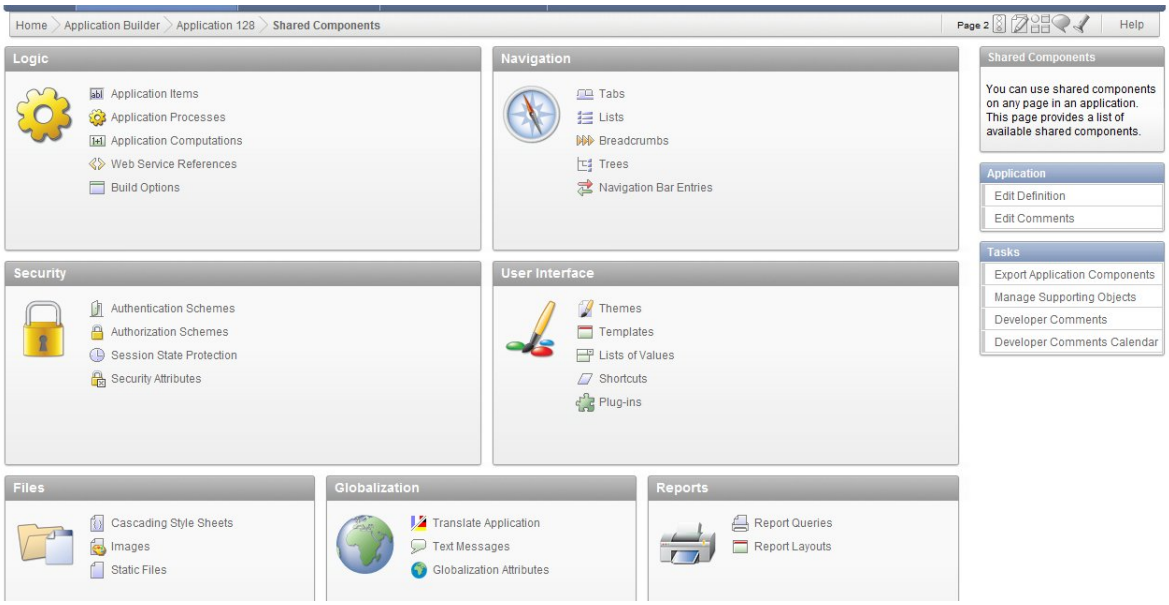
Voor het invullen van de tabel **weerstations_gemeenten** zijn er twee procedures en één functie die gebundeld zijn een *package* (broncode C.13). De functie **haversinedistance** heeft als inputparameters

de lengte- en breedtegraad van twee locaties. De afstandsmaat die berekend wordt, houdt rekening met de bolvorm van het vlak waarop de coördinaten zich bevinden. Deze functie wordt gebruikt in de procedure **EersteDrie**. De parameter die moet opgegeven worden is het volgnummer van een gemeente uit de tabel **gemeenten**. De afstanden tussen de gemeente en de weerstations uit de *view* **beschikbare_weerstations** worden berekend en ingevuld in **tijdelijk_ws_gem**. De tabel wordt geordend op de berekende afstand en de eerste drie worden in de tabel **weerstations_gemeenten** ingevuld. De procedure **UpdateWeerstatGem** selecteert de gemeenten uit de tabel **gemeenten** die nog niet ingevuld zijn in **weerstations_gemeenten**. Per gemeente wordt dan de procedure **EersteDrie** uitgevoerd. Deze procedure kan uitgevoerd worden indien er bijvoorbeeld nieuwe weerstations zijn bijgekomen en men alle afstanden wil herberekenen.

3.4.3 Authenticatie gebruiker

Authenticatie is het mechanisme dat de identiteit van de gebruiker controleert en deze toegang verleent tot een systeem of applicatie gebaseerd op de gebruikersnaam en wachtwoord die opgegeven worden (12). Dit mechanisme is niet te verwarren met autorisatie waarbij het niveau van toegang en gebruik tot het systeem wordt beheerd (bijvoorbeeld bij verschillende gebruikersrollen). De ontwikkelaar kan beide schema's beheren in de *shared components* van een applicatie (zie Figuur 3.3). De toegangscontrole of authenticatie in een applicatie kan in APEX op vier manieren gebeuren.

- Geen authenticatie, waarbij er om het even welke gebruiker toegang tot het systeem kan krijgen. Dit is het geval bij andere applicaties van de BDB. Bij het publieke gedeelte van deze applicatie is er ook geen authenticatie nodig.
- De standaardauthenticatiemethodes voor de gebruikers van de APEX-omgeving. Dit systeem is echter niet erg schaalbaar aangezien de beheerder van de APEX-omgeving elke gebruiker individueel moet aanmaken. Bij het authenticatieschema moet in dit geval in het *Authentication Function*-veld het sleutelwoord `-BUILTIN-` ingegeven worden.
- De *login-credentials* van de individuele databankgebruikers kan men gebruiken door in het *Authentication Function*-veld het sleutelwoord `-DBACCOUNT-` in te geven.
- De functies om de gebruikersgegevens te beheren en controleren kan men ook uit een zelfgedefinieerde tabel en schema halen. De BDB gebruikt deze methode bij de applicaties die niet publiek zijn op basis van de gebruikersgegevens in de tabel *web_prod*. **web_login**. In deze applicatie wordt de functies uit het pakket **app_security_pkg** (broncode C.12) gebruikt voor het controleren van de gebruikersgegevens (8).



Figuur 3.3: Bij de *shared components* vindt men de onderdelen van de applicatie die over de gehele applicatie gebruikt kunnen worden. Het gaat hier om zaken zoals navigatie-elementen, applicatie-items en authenticatie- en authorisatieschema's.

3.4.4 Sessie-objecten en navigatie-elementen

De navigatie in de applicatie gebeurt hoofdzakelijk via een tabs aan de bovenkant van de pagina's. Deze worden bij de definitie van een pagina toegevoegd of kunnen beheerd worden bij de *shared components*. De tabs bestaat uit twee niveau's. Op het eerste niveau wordt er onderscheid gemaakt tussen het publieke gedeelte (tabset1: Hoofdpagina) en de gedeelte van de site waarvoor ingelogd moet worden (tabset2: Beheer van percelen). Onder deze twee tabset worden de tabs verdeeld die respectievelijk bij het publieke en het niet-publieke deel van de applicatie horen. Bij één tab kunnen ook verschillende pagina's van de applicatie horen. In dat geval wordt een derde niveau van navigatie toegevoegd als een lijst die aan de zijkant van de pagina wordt getoond als de gebruiker zich op een van de pagina's van deze tab bevindt. Dit kan men bereiken door de lijst toe te voegen aan de *page zero* van de applicatie en als conditie bij het tonen van deze regio in te stellen dat deze enkel mag verschijnen op de pagina's van deze tab. Uiteindelijk is er aan de *page zero* een *breadcrumb* toegevoegd die toont op welke pagina en op welk navigatieniveau van de applicatie de gebruiker zich bevindt. De organisatie van de tabsets en tabs is als volgt:

- Tabset1: Hoofdpagina
  - Tab: Hoofdpagina
    - Pagina 1: Hoofdpagina - defaultpagina
  - Tab: Algemene informatie over *Fusarium*
    - Pagina 102:. Pathogenen - defaultpagina
    - Pagina 103: Symptomen
    - Pagina 104: Levenscyclus
    - Pagina 105: Gevolgen

        * Pagina 106: Invloedsfactoren
    - Tab: Veldproeven van Hogeschool Gent
        * Pagina 107: Proefvelden - defaultpagina
        * Pagina 108: Visuele aantastingen
        * Pagina 109: DON bepalingen
        * Pagina 110: Species bepalingen
    - Tab: Nuttige Links
        * Pagina 4: Nuttige Links en regelgeving - defaultpagina
    - Tab: Kaart met resultaten
        * Pagina 6: Kaart met resultaten - defaultpagina

- Tabset2: Beheer van percelen
    - Tab: Beheer van percelen
        * Pagina 5: Beheer van percelen - defaultpagina
        * Pagina 2: Voeg perceel toe
    - Tab: Jaarlijkse Gegevens
        * Pagina 7: Jaarlijkse Gegevens - defaultpagina
        * Pagina 12: Voeg jaar toe
    - Tab: Voorspellingen
        * Pagina 13: Voorspellingsrapporten - defaultpagina

Op de login-pagina van de applicatie wordt standaard volgens zijn *template* geen navigatieelementen getoond zoals de tabs en *breadcrumbs*. Er zijn drie applicatie-items bij de *shared components* gedefinieerd.

- AI_PRT_CONTACTEN_NR: Hierin wordt na het inloggen het volgnummer uit *prtdev.***prt_contacten** ingevuld dat overeenkomt met de ingelogde gebruiker in *web_prod.***web_login**. Dit is nodig om de juiste percelen uit *web_prod.***web_percelen** te tonen bij het inloggen. Hiervoor zijn namelijk de rechten gedefinieerd in de tabellen **web_bedrijven_contacten** en **web_rechten** op basis van dit contactenvolgnummer.

- F128_PERCEEL: Uit de percelen die de gebruiker getoond worden (en waarvoor hij dus de rechten heeft om deze te zien) kan er één geselecteerd worden om de groeiseizoenen te bekijken. Het volgnummer van dit perceel uit *web_prod.***web_percelen** wordt in dit applicatie-item ingevuld bij selectie van een perceel. Als dit nog niet ingevuld is, wordt het tab van de Jaarlijkse Gegevens nog niet getoond. Als de gebruiker een perceel heeft bekeken en dan bijvoorbeeld terugkeert naar het beheer van de percelen om een nieuw perceel toe te voegen, blijft het tab met het bekeken perceel openstaan en kan hij hier later naar terugkeren om er verder aan te werken.

- F128_JAAR: Voor de groeiseizoenen kan de gebruiker ervoor kiezen een van deze te selecteren om de voorspellingsrapporten te tonen. Hierbij wordt het volgnummer van het geselecteerde groeiseizoen in **perceel_jaren** ingevuld in dit applicatie-item. Als deze nog niet ingevuld is, is het tab van de voorspellingen ook niet zichtbaar in de applicatie. Als de gebruiker een ander perceel bekijkt wordt de waarde in dit item verwijderd zodat er geen tab met voorspellingsrapporten blijft openstaan voor een groeiseizoen van een ander perceel.

Uiteindelijk is er aan de *page zero* van een applicatie een regio toegevoegd die gebaseerd is op de *view* **rapporten_vandaag**. Hierdoor kan de gebruiker over de gehele applicatie opvolgen wat de status is van alle voorspellingsrapporten die diezelfde dag zijn opgevraagd.

3.4.5 Pagina's

Bij het opstarten van de applicatie komt de gebruiker op de hoofdpagina van de applicatie terecht. Ook als de gebruiker uitlogt komt hij terug op deze pagina terecht. Deze pagina bevat een korte introductie van het project dat tot het voorspellingsmodel heeft geleid en de partners van het onderzoek. Daarnaast is er in het publieke gedeelte van de applicatie nog de tab met algemene informatie die de gebruiker introduceert tot de algemene principes van de *Fusarium*-problematiek. Onder het tab van de veldproeven van de Hogeschool Gent wordt meer informatie gegeven over de resultaten van de veldproeven die werden uitgevoerd in kader van het project. Bij de nuttige links zijn er informatie en verwijzingen te vinden die de gebruiker informeren over wettelijk verplichtingen waaraan in verband met *Fusarium* moet voldaan zijn. Uiteindelijk is er in het publieke gedeelte een tab met de AnyChart-kaart gebaseerd op de *view* **kaart**. Bij deze kaart kan de gebruiker door de resultaten van de verschillende jaren van het onderzoek bladeren en dit voor enerzijds de gemiddelde ziekte-index per arrondissement en anderzijds voor het gemiddeld DON-gehalte per arrondissement.

Als de gebruiker naar de tabset voor het beheer van percelen navigeert, wordt hij naar de login-pagina verwezen als hij nog niet ingelogd is. Na het inloggen komt de gebruiker terecht op het beheer van de percelen waartoe hij rechten heeft als landbouwer of gewasadviseur om deze te bekijken. Er is een overzicht van de percelen en een knop die verwijst naar de pagina voor het toevoegen van een nieuw perceel. Bij het toevoegen van het perceel is het belangrijkste dat de correcte gemeente wordt opgegeven voor de koppeling met de weerstations. Dit is dan ook een verplicht veld in het formulier. De gebruiker geeft zijn postnummer in en daar bij wordt via een *dynamic action* de overeenkomstige gemeente(n) ingeladen in de selectielijst voor de gemeente(14). Na het toevoegen van een perceel keert de applicatie terug naar het overzicht van de percelen. Per perceel is er ook een icoon waarop de gebruiker kan klikken om de groeiseizoenen van dat perceel te bekijken.

De groeiseizoenen zijn volgens jaar in dalende volgorde gesorteerd zodat het meest recente en dus meest relevante jaar bovenaan staat. De pagina bevat ook drie koppen. De eerste ("voeg nieuw jaar toe") is voor een nieuw groeiseizoen aan te maken en is enkel zichtbaar als er voor dat jaar (volgens de regels voor het invullen van jaar bij de trigger van de tabel **perceel_jaren**) nog geen groeiseizoen is aangemaakt. Bij het aanmaken van een nieuw groeiseizoen wordt gevraagd om alle variabelen verplicht in te vullen die nodig zijn voor het voorspellingsmodel. Voor de oogstdatum kan het hier dus om een schatting van de gebruiker zijn. Als het nieuw jaar is toegevoegd keert de applicatie terug naar het overzicht van de jaren. Als dit jaar is toegevoegd worden de twee andere knoppen zichtbaar. Enerzijds kan de landbouwer in de loop van het jaar zijn gegevens aanpassen die nodig zijn via de knop "pas aan" die op dezelfde pagina een regio laadt met een formulier. Als op de knop "voeg toe" wordt gedrukt kan de landbouwer meewerken aan het onderzoek door een observatie van de bloeidatum, fungicidebehandeling of ziekt-index te doen. Het activeren van één regio horende bij de knop "pas aan" of "voeg toe" op de pagina verbergt automatisch de andere regio. Ook bij de groeiseizoenen is er een icoontje per groeiseizoen in het overzicht om door te klikken naar het overzicht van de voorspellingsrapporten van dat groeiseizoen.

Naast het overzicht van de voorspellingsrapporten is er ook een knop die toelaat om een nieuw voorspellingsrapport op te vragen. Deze knop is echter enkel zichtbaar bij het huidige groeiseizoen zodat

de gebruiker verhinderd wordt in het opvragen van groeiseizoenen die afgelopen zijn. De historische voorspellingsrapporten blijven echter wel zichtbaar voor de gebruiker. Bij de voorspellingsrapporten kan de gebruiker opnieuw klikken op een icoontje om de voorspelde waarden voor de verschillende scenario's te bekijken. Hierdoor wordt een regio op dezelfde pagina geladen met de voorspellingen in tabel- en grafiekvorm.

Hoofdstuk 4

Besluit

Het databankschema en de website die in het kader van deze masterproef werden ontwikkeld bevatten de basis van de gewenste functionaliteit en bieden een raamwerk voor de verdere implementatie van de gewenste functionaliteiten. Tijdens de paar maanden die het werk voor deze masterproef besloeg, werd van de functionele analyse en analyse van de bestaande omgevingen naar de uiteindelijke implementatie van een groot deel van de webtool gegaan. Dit was deels te danken aan de reeds aanwezige tabelstructuren van het project *Fusarium* en de BDB evenals de BDBnet-authenticatie die voor de toegang tot de andere internet-diensten van de BDB wordt gebruikt. Daarnaast maakte de ontwikkeling van de webtool binnen de ontwikkelingsomgeving APEX het mogelijk om snel de functionaliteit te bouwen die binnen het raamwerk van deze ontwikkelingsomgeving past. Ook het gebruik van PL/SQL als *serverside* scriptingtaal neemt een extra barrière weg voor de ontwikkelaar omdat bij de ontwikkeling in een andere programmeertaal ook ontwikkeling en tijd zou moeten besteed worden aan een databankabstractielaag terwijl in dit geval de gegevens in de databank rechtstreeks kunnen gemanipuleerd worden. Dit neemt niet weg dat doordat er al zoveel functionaliteit beschikbaar is via de ontwikkelingsomgeving, het moeilijk wordt om als ontwikkelaar buiten de lijnen te kleuren. Verder vergt het gebruik van PL/SQL als programmeertaal voor dergelijke applicaties enig aanpassingsvermogen van een ontwikkelaar. Deze masterproef dient dan ook als leidraad voor toekomstige ontwikkeling van de webtool.

Bijlage A

Opzet van de ontwikkelingsomgeving

A.1 Lokaal opzetten van Oracle DB 10g en APEX 4.0

1. Voor het afladen van de *executables* van Oracle DB en APEX moet een account aangemaakt worden op de website van Oracle.
2. De gratis versie van Oracle DB is Oracle DB 10g Express Edition: download en installeer.
3. Ga naar de Hoofdpagina (*Go To Database Home Page*) van het geïnstalleerde databanksysteem (verzeker hierbij dat het databank is opgestart (*Start Database*)). Log in als de gebruiker "system" en met het wachtwoord dat opgegeven werd bij de installatie. Deze versie gebruikt de oudere versie van APEX. Om over te stappen naar APEX 4.0 volg volgende stappen.
4. Download versie 4.0 van de website van Oracle hier.
5. Pak de gecomprimeerde map met APEX uit.
6. Navigeer via de command-shell naar de uitgepakte map en voer volgende commando's uit om de *environment*- en *login*-variabelen juist in te stellen:

   ```
   set ORACLE_HOME=\map\met\executables\van\oracle\server
   set ORACLE_SID=XE
   set PATH=%ORACLE_HOME%\bin;%PATH%
   ```

   Als de installatie in Windows gebeurt moet de ORACLE_HOME-variable op volgende waarde gezet worden:

   ```
   c:\oraclexe\app\oracle\product\10.2.0\server
   ```

7. Log in via de command-line als de databankadministrator in de command-line-interface van de databank met volgend commando:

   ```
   sqlplus "/ as sysdba"
   ```

8. Om de installatie te starten voor volgend commando uit in de SQL-omgeving:

   ```
   @apexins SYSAUX SYSAUX TEMP /i/
   ```

9. Na de installatie moeten de afbeeldingen voor de ontwikkelingsomgeving ook nog ingeladen worden door het volgende commando uit te voeren in de SQL-omgeving:

   ```
   @apxldimg \map\waar\Apex\is\in\uitgepakt
   ```

10. Om het wachtwoord van de admin-account aan te passen moet volgend commando in de SQL-omgeving uitgevoerd worden:

    `@apxxepwd.sql`

11. Om de SQL-omgeving te verlaten gebruik het commando `exit`.
12. Om in de administratie-omgeving van APEX te komen navigeer de browser naar

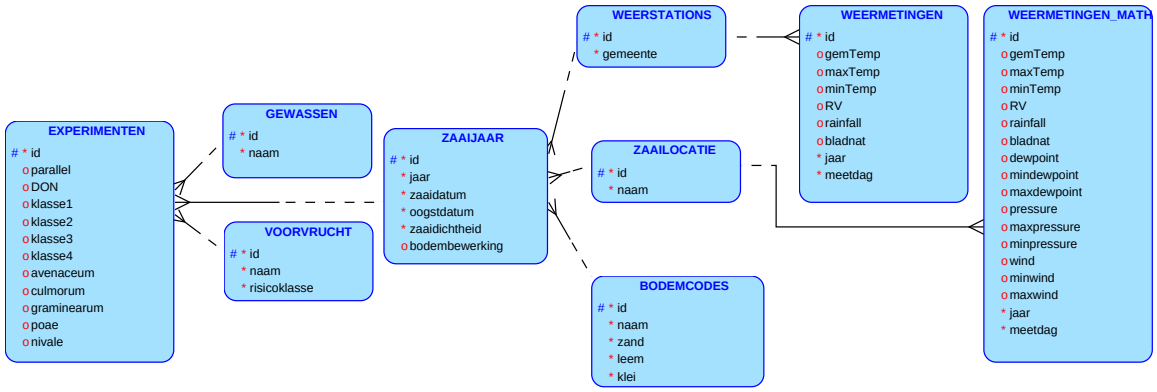
    http://127.0.0.1:8080/apex/apex_admin.

A.2 Toegang tot de ontwikkelingsomgeving van de BDB

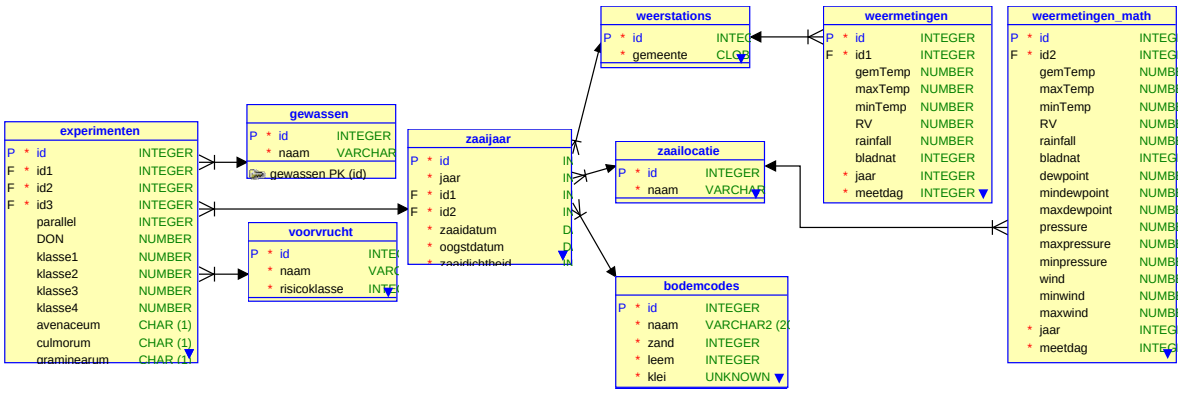
1. Netwerkcentrum > Verbinding of netwerk instellen > Verbinding met werkplek maken > Nieuwe verbinding > Mijn internetverbinding (VPN) gebruiken
2. Internetadres: mail.bdb.be
3. Gebruiker en wachtwoord bij netwerkbeheerder van de bodemkundige dienst te verkrijgen.
4. Klik op Eigenschappen en configureer de overige opties voor de verbinding:
    - ga naar tabblad "Netwerken"
    - dubbelklikken "Internet protocol versie 4 (TCP/IP v4)" uit lijst (niet deselecteren): er wordt een schermpje met "eigenschappen" geladen.
    - "De volgende DNS-serveradressen gebruiken:" selecteren. Vul als voorkeur-DNS-server in 172.16.164.118
    - klik in schermpje eigenschappen op geavanceerd: er wordt een schermpje met "Geavanceerde eigenschappen" geladen.
    - Deselecteer "Standaard-gateway in netwerk gebruiken"
5. Om van op het UGent-netwerk met het VPN-netwerk van de BDB verbinden te leggen, moet dit aangevraagd worden bij de helpdesk van de UGent. Hiervoor moet een gemotiveerde email (bvb. voor mijn thesis) naar de helpdesk gestuurd worden, waarbij ook vermeld wordt wat het eigen vast UGent IP-adres is, het IP of de hostnaam van de VPN-gateway (mail.bdb.be dus) en welk type van VPN (PTPP met standaard windows-VPN-client).
6. Na het opstarten van de VPN navigeer naar de APEX-ontwikkelingsomgeving van de BDB.
7. Log in op de *workspace* "Fusarium" met uw gebruikersnaam en wachtwoord (ook via de netwerkbeheerder van de BDB te verkrijgen).

Bijlage B

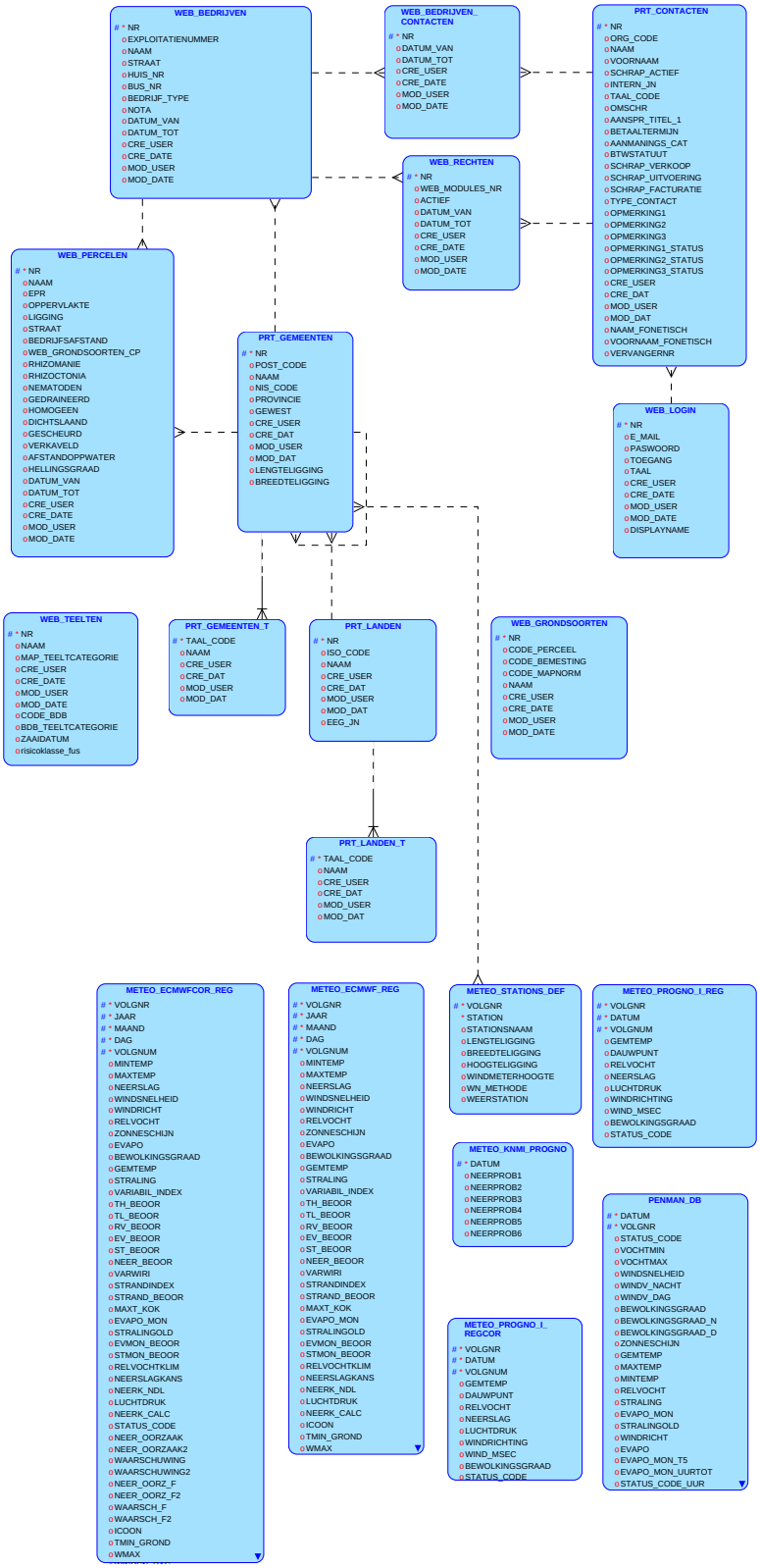
ER-diagrammen



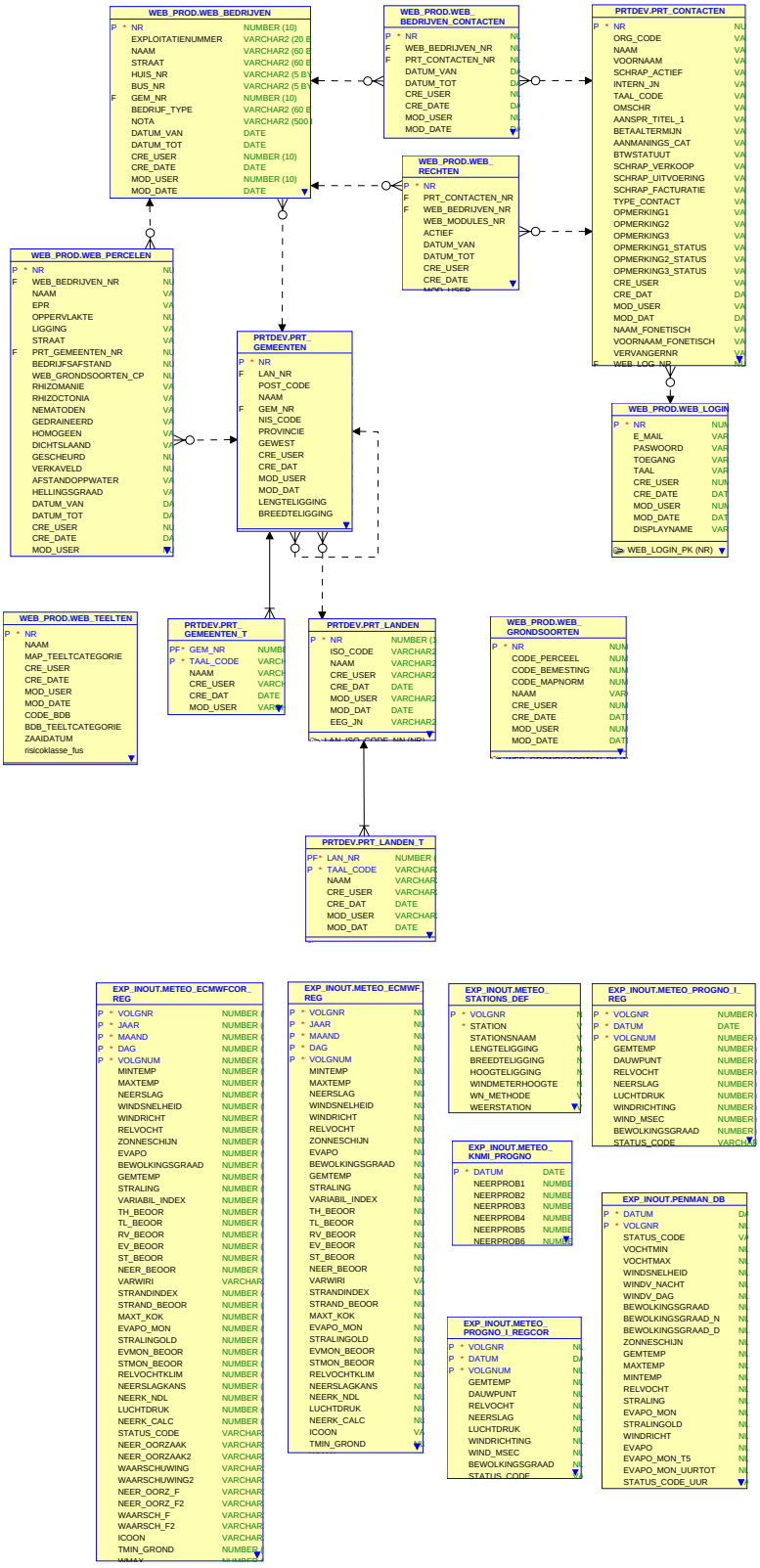
Figuur B.1: Logisch ontwerp databank van project *Fusarium*



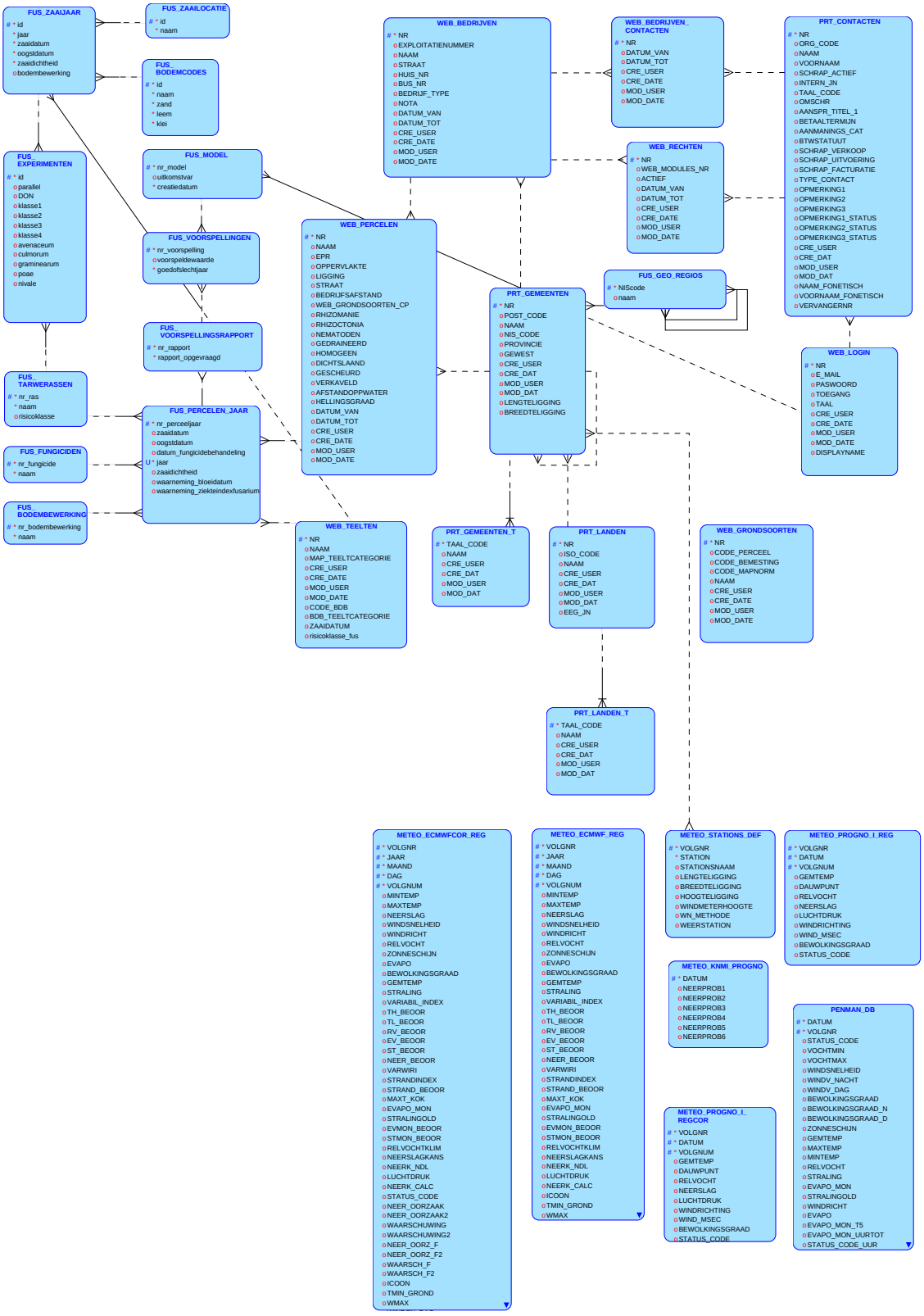
Figuur B.2: Logisch ontwerp databank van project *Fusarium*



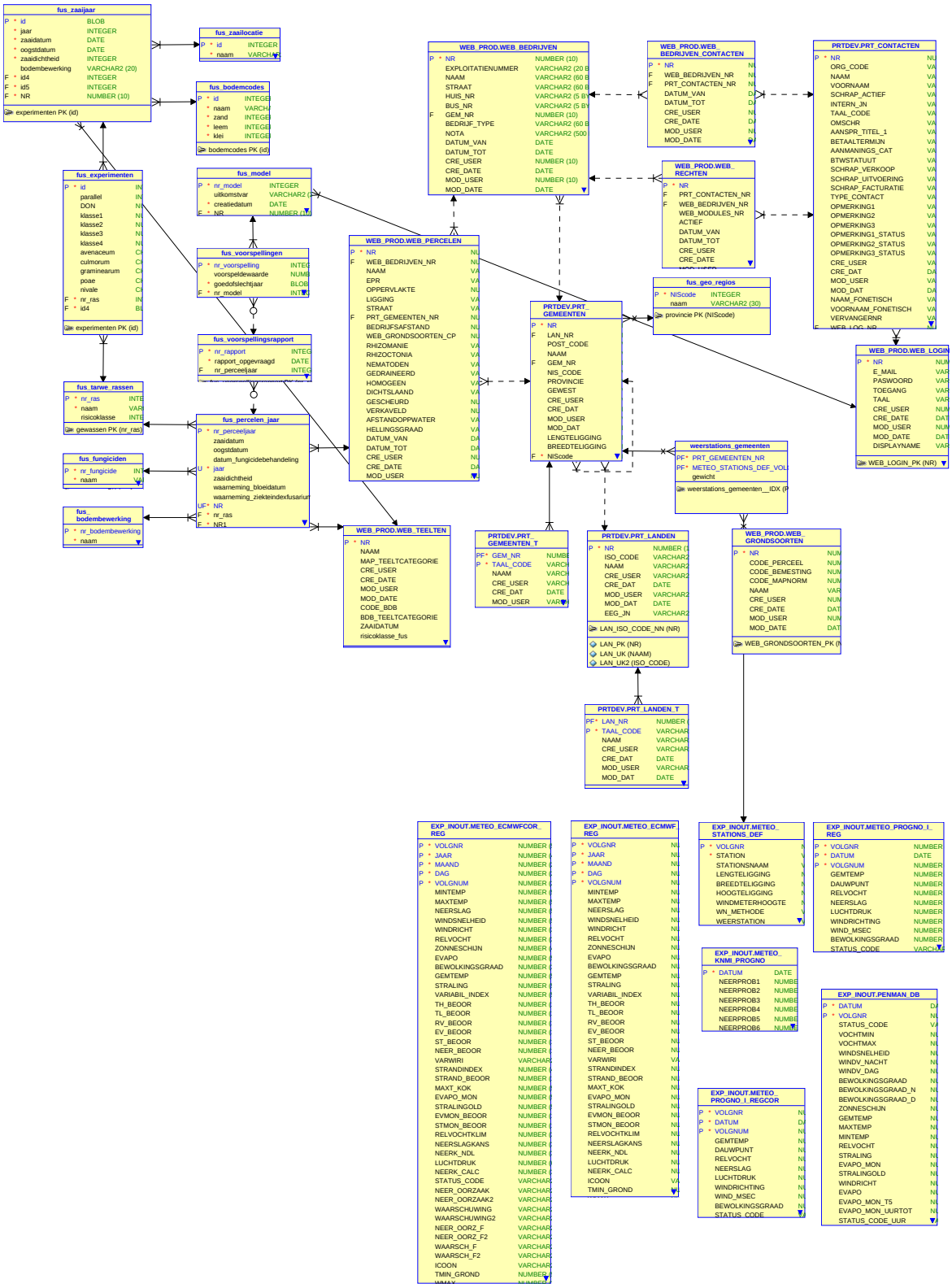
Figuur B.3: Logisch ontwerp databank van BDB



Figuur B.4: Relationeel ontwerp databank van BDB.



Figuur B.5: Logisch ontwerp databank van BDB na uitbreiding met nieuwe entiteiten en samenvoeging met andere databank



Figuur B.6: Relationeel ontwerp databank van BDB na uitbreiding met nieuwe entiteiten en samenvoeging met andere databank

Bijlage C

Broncodes

C.1 Views

Broncode C.1: samenvattingexperimenten

```
CREATE OR REPLACE FORCE VIEW fusarium.samenvattingexperimenten
(
        zaaijaar_id ,
        aanwezigheid_don_species ,
        zi ,
        don)
AS
SELECT
        zaaijaar_id ,
        DECODE ((graminearum || culmorum), 'ff', 'Neen', 'Ja'),
        100 * (1 * klasse2 + 2 * klasse3 + 3 * klasse4 + 4 * klasse5) / 400,
        don
FROM fusarium.experimenten
WITH READ ONLY;
```

Broncode C.2: kaart

```
CREATE OR REPLACE FORCE VIEW fusarium.kaart (naam, jaar, "DON-gehalte", zi)
AS
SELECT
        gg.naam, zj.jaar,
        ROUND (AVG (don), 4) "DON-gehalte",
        ROUND (AVG (zi), 4)
FROM
        fusarium.samenvattingexperimenten EXP INNER JOIN
        fusarium.zaaijaar zj ON (EXP.zaaijaar_id = zj.ID) INNER JOIN
        fusarium.zaailocaties zl ON (zj.zaailocatie_id = zl.ID)INNER JOIN
        fusarium.gemeenten g ON (UPPER (zl.naam) LIKE g.naam) INNER JOIN
        fusarium.gemeenten_grp gg ON (TRUNC (nis_code, -3) / 1000 = code)
WHERE code IS NOT NULL
```

```
GROUP BY zj.jaar, gg.naam
WITH READ ONLY;
```

Broncode C.3: beschikbare_weerstations

```
CREATE OR REPLACE FORCE VIEW fusarium.beschikbare_weerstations
(       volgnr,
        lengte,
        breedte)
AS
SELECT
        volgnr,
         lengteligging / 10000 lengte,
         breedteligging / 10000 breedte
FROM exp_inout.meteo_stations_def
WHERE volgnr IN (
        SELECT DISTINCT (volgnr)
        FROM exp_inout.penman_db tb
        WHERE TO_NUMBER(EXTRACT (YEAR FROM tb.datum)) > 2000)
        AND (lengteligging != 0 AND breedteligging != 0)
WITH READ ONLY;
```

Broncode C.4: maandelijksewaarden

```
CREATE OR REPLACE FORCE VIEW fusarium.maandelijksewaarden
(       vochtminmin,
        vochtmaxmax,
        windsnelheidgem,
        bewolkingsgraadgem,
        zonneschijngem,
        gemgemtemp,
        minmintemp,
        maxmaxtemp,
        mediaanrelvocht,
        jaar,
        maand,
        volgnr)
AS
SELECT
        MIN (vochtmin),
        MAX (vochtmax),
        AVG (windsnelheid),
        AVG (bewolkingsgraad),
        AVG (zonneschijn),
        AVG (gemtemp),
        MIN (mintemp),
        MAX (maxtemp),
        MEDIAN (relvocht),
```

```
        EXTRACT (YEAR FROM tb.datum) jaar,
        EXTRACT (MONTH FROM tb.datum) maand,
        volgnr
FROM
        exp_inout.penman_db tb
GROUP BY
        EXTRACT (YEAR FROM tb.datum),
        EXTRACT (MONTH FROM tb.datum),
        volgnr
HAVING
        EXTRACT (YEAR FROM tb.datum) > 2000
ORDER BY
        jaar,
        maand,
        volgnr;
```

Broncode C.5: beschikbaremodellen

```
CREATE OR REPLACE FORCE VIEW fusarium.beschikbaremodellen
(
        d,
        r)
AS
SELECT
        em.uitkomstvar d,
        nr_model r
FROM
        model em,
        (SELECT
                uitkomstvar,
                MAX (creatiedatum) datem
        FROM model
        GROUP BY uitkomstvar) m2
WHERE
        em.uitkomstvar = m2.uitkomstvar
        AND em.creatiedatum = m2.datem;
```

Broncode C.6: status_rapporten

```
CREATE OR REPLACE FORCE VIEW fusarium.status_rapporten
(       percentage,
        nr_rapport)
AS
SELECT
        100 * SUM (NVL2 (voorspeldewaarde, 1, 0)) / COUNT (*) percentage,
        nr_rapport
FROM voorspellingen
GROUP BY nr_rapport;
```

Broncode C.7: rapporten_vandaag

```
CREATE OR REPLACE FORCE VIEW fusarium.rapporten_vandaag
(       perceelnaam,
        rapport_opgevraagd,
        status,
        connr)
AS
SELECT
        naam perceelnaam,
        rapport_opgevraagd,
        percentage || '%' status,
        wb.connr
FROM voorspellingsrapport NATURAL JOIN
        status_rapporten NATURAL JOIN
        percelen_jaar pj INNER JOIN
        web_prod.web_percelen wp USING (nr) INNER JOIN
        (
                (
                SELECT
                        c.nr connr,
                        wb.naam bedrijf,
                        wb.nr bednr
                FROM prtdev.prt_contacten c INNER JOIN
                        web_prod.web_bedrijven_contacten wbc
                        ON (c.nr = wbc.prt_contacten_nr) INNER JOIN
                        web_prod.web_bedrijven wb
                        ON (wbc.web_bedrijven_nr = wb.nr)
                )
        UNION ALL
                (
                SELECT
                        c.nr connr,
                        wb.naam bedrijf,
                        wb.nr bednr
                FROM prtdev.prt_contacten c INNER JOIN
                        web_prod.web_rechten wbc
                        ON (c.nr = wbc.prt_contacten_nr) INNER JOIN
                        web_prod.web_bedrijven wb
                        ON (wbc.web_bedrijven_nr = wb.nr)
                )
        ) wb ON (wb.bednr = wp.web_bedrijven_nr)
WHERE
        TO_CHAR (rapport_opgevraagd, 'dd-mon-yy') =
        TO_CHAR(CURRENT_DATE, 'dd-mon-yy');
```

C.2 Triggers

Broncode C.8: perceeljaar_trigger

```
CREATE OR REPLACE TRIGGER FUSARIUM.PERCEELJAAR_TRIGGER
BEFORE INSERT
ON FUSARIUM.PERCELEN_JAAR
REFERENCING NEW AS nieuw OLD AS oud
FOR EACH ROW
BEGIN
        IF :nieuw.nr_perceeljaar IS NULL THEN
                SELECT perceeljaar_rij.nextval
                INTO :nieuw.nr_perceeljaar
                FROM dual;
        END IF;
        SELECT
                (CASE
                        WHEN EXTRACT(MONTH FROM Current_Date)>8
                                THEN (EXTRACT(YEAR FROM Current_Date)+1)
                                ELSE EXTRACT(YEAR FROM Current_Date)
                END)
        INTO :nieuw.jaar FROM dual;
END;
```

Broncode C.9: model_trigger

```
CREATE OR REPLACE TRIGGER FUSARIUM.MODEL_TRIGGER
BEFORE INSERT
ON FUSARIUM.MODEL REFERENCING NEW AS nieuw OLD AS oud
FOR EACH ROW
BEGIN
        IF :nieuw.nr_model IS NULL THEN
                SELECT modelrij.nextval
                INTO :nieuw.nr_model FROM dual;
        END IF;
        SELECT CURRENT_DATE
        INTO :nieuw.creatiedatum FROM dual;
END;
```

Broncode C.10: voorspellingsrapport_trigger

```
CREATE OR REPLACE TRIGGER FUSARIUM.voorspellingsrapport_trigger
BEFORE INSERT
ON FUSARIUM.VOORSPELLINGSRAPPORT REFERENCING NEW AS nieuw
FOR EACH ROW
BEGIN
        IF :nieuw.nr_rapport IS NULL THEN
                SELECT voorspellingsrapport_rij.nextval
                INTO :nieuw.nr_rapport FROM dual;
        END IF;
        IF :nieuw.rapport_opgevraagd IS NULL THEN
```

```
                SELECT CURRENT_DATE
                INTO :nieuw.rapport_opgevraagd FROM dual;
        END IF;
END;
```

Broncode C.11: voorspelling_trigger

```
CREATE OR REPLACE TRIGGER FUSARIUM.VOORSPELLING_TRIGGER
BEFORE INSERT
ON FUSARIUM.VOORSPELLINGEN
REFERENCING NEW AS nieuw OLD AS oud
FOR EACH ROW
DECLARE
        voorspelling NUMBER;
BEGIN
        IF :nieuw.nr_voorspelling IS NULL THEN
                SELECT voorspelling_rij.nextval
                INTO :nieuw.nr_voorspelling FROM dual;
        END IF;
        SELECT CURRENT_DATE
        INTO :nieuw.voorspelling_opgevraagd FROM dual;
        --Doe voorspelling
        SELECT dbms_random.value(0,100)/100
        INTO :nieuw.voorspeldewaarde FROM dual;
EXCEPTION
        WHEN OTHERS THEN RAISE;
END VOORSPELLING_TRIGGER;
```

C.3 Procedures en functies

Broncode C.12: app_security_pkg

```
CREATE OR REPLACE PACKAGE BODY FUSARIUM.app_security_pkg
AS

        PROCEDURE login
                (p_uname IN VARCHAR2
                ,p_password IN VARCHAR2
                , p_session_id IN VARCHAR2
                ,p_flow_page IN VARCHAR2)
        IS
                lv_goto_page NUMBER DEFAULT 5;
        BEGIN
                wwv_flow_custom_auth_std.login
                (p_uname => p_uname,
                p_password => p_password,
                p_session_id => p_session_id,
                p_flow_page => p_flow_page || ':' || lv_goto_page
```

```
                );
        EXCEPTION
                WHEN OTHERS
                THEN RAISE;
        END login;

        PROCEDURE valid_user2
                (p_username IN VARCHAR2,
                p_password IN VARCHAR2)
        AS
                v_dummy VARCHAR2 (1);
        BEGIN
                SELECT '1'
                INTO v_dummy
                FROM WEB_PROD.WEB_LOGIN
                WHERE
                        UPPER (WEB_PROD.WEB_LOGIN.E_MAIL)
                        = UPPER (p_username)
                        AND WEB_PROD.WEB_LOGIN.PASWOORD
                        = p_password;
        EXCEPTION
        WHEN NO_DATA_FOUND
                THEN  raise_application_error   (
                -20000,
                ' Invalid  username / password. ' );
        END valid_user2;

        FUNCTION valid_user
                (p_username IN VARCHAR2,
                p_password IN VARCHAR2)
        RETURN BOOLEAN
        AS
        BEGIN
                 valid_user2 (UPPER(p_username), p_password);
                RETURN TRUE;
        EXCEPTION
                WHEN OTHERS
                THEN RETURN FALSE;
        END valid_user;

END app_security_pkg;
/
```

Broncode C.13: linken_weerstations_gemeenten

```
CREATE OR REPLACE PACKAGE BODY FUSARIUM.linken_weerstations_gemeenten
AS
```

```
FUNCTION haversinedistance
        (lengte1  IN NUMBER,
        breedte1  IN NUMBER,
        lengte2  IN NUMBER,
        breedte2  IN NUMBER)
-- gebaseerd op http://en.wikipedia.org/wiki/Haversine_formula
-- (houdt geen rekening met geoïde vorm van de aarde)
RETURN NUMBER
IS
        dist  NUMBER; -- kilometers
        Pi  CONSTANT NUMBER := 3.141592653589793; -- geen eenheid...
        EarthRad CONSTANT INTEGER := 6371; -- in kilometers
BEGIN
        dist  :=
        EarthRad*2*Asin(
                Sqrt(
                        Power(Sin((breedte1  - breedte2)*Pi  /180/2),2)  +
                        Cos(breedte1*Pi/180)*Cos(breedte2*Pi/180)
                        *Power(Sin((lengte1  - lengte2)*Pi  /180/2),2)
                        )
                );
        RETURN dist;
END haversinedistance;

PROCEDURE EersteDrie
        (NR IN NUMBER)
IS
        CURSOR cur IS
        SELECT *
        FROM beschikbare_weerstations;
        Nummer CONSTANT NUMBER := NR;
        l  NUMBER;
        b  NUMBER;
        table_max  PLS_INTEGER;
BEGIN
        DELETE FROM tijdelijke_ws_gem;
        table_max  :=  0;
        SELECT
                TO_NUMBER(fs.lengte),
                TO_NUMBER(fs.breedte)
        INTO
                l,
                b
        FROM Fusarium.gemeenten fs
        WHERE fs.NR=Nummer;
        --dbms_output.put_line(l ||'    '|| b);
        FOR ws IN cur LOOP
```

```
                        INSERT INTO tijdelijke_ws_gem
                        VALUES
                                (Nummer,
                                ws.volgnr,
                                Fusarium.haversinedistance
                                        (ws.lengte,
                                        ws.breedte,l,b)
                                );
                END LOOP;
                INSERT INTO Fusarium.weerstations_gemeenten
                (
                SELECT *
                FROM
                        (SELECT * FROM tijdelijke_ws_gem
                        WHERE GEM_NR = Nummer
                        ORDER BY GEW ASC
                        )
                WHERE ROWNUM < 4);
                COMMIT;
        EXCEPTION
                WHEN OTHERS THEN
                  raise_application_error (-20001,
                'An error was encountered - ' || SQLCODE||' -ERROR- '||SQLERRM);
        END EersteDrie;

        PROCEDURE UpdateWeerstatGem
        IS
                CURSOR cur IS
                SELECT nr
                FROM gemeenten
                WHERE
                        lengte IS NOT NULL
                        AND breedte IS NOT NULL
                        AND NR NOT IN (
                                SELECT distinct(gemeenten_nr)
                                FROMweerstations_gemeenten);
        BEGIN
                FOR ws IN cur LOOP
                        EersteDrie(ws.NR);
                END LOOP;
        EXCEPTION
                WHEN OTHERS THEN
                  raise_application_error (-20001,
                'An error was encountered - ' || SQLCODE||' -ERROR- '||SQLERRM);
        END UpdateWeerstatGem;

END linken_weerstations_gemeenten;
```

Broncode C.14: Voegvoorspellingtoe

```
CREATE OR REPLACE PROCEDURE FUSARIUM.Voegvoorspellingtoe
(
        NR_JAAR IN NUMBER,
        NR_MOD IN NUMBER
)
IS
        NR_RAP NUMBER;
BEGIN
        --Aanmaak voorspellingsrapport

        SELECT  voorspellingsrapport_rij  . nextval
        INTO NR_RAP FROM dual;
        INSERT INTO  voorspellingsrapport
                (NR_RAPPORT,
                NR_PERCEELJAAR)
        VALUES
                (NR_RAP,
                NR_JAAR);
        COMMIT;

        --Voeg voorspellingen toe aan voorspellingsrapporten
        --(verschillende modellen mogelijk in een rapport)

        INSERT INTO voorspellingen (
                NR_RAPPORT,
                NR_MODEL,
                GOEDOFSLECHTJAAR)
        VALUES (
                NR_RAP,
                NR_MOD,
                'g' );
        COMMIT;
        INSERT INTO voorspellingen (
                NR_RAPPORT,
                NR_MODEL,
                GOEDOFSLECHTJAAR)
        VALUES (
                NR_RAP,
                NR_MOD,
                's' );
        COMMIT;

EXCEPTION
        WHEN NO_DATA_FOUND THEN
```

```
    NULL;
  WHEN OTHERS THEN
    RAISE;
END Voegvoorspellingtoe ;
```

Bijlage D

Inhoud van de bijgevoegde CD

- Masterproef_CD
  - Analyses
    - UML ..... Secties 3.1 en 3.2
    - ER ..... Secties 3.3 en 3.4.1 en Bijlage B
  - Implementatie
    - SQL-Code ..... Secties 3.4.1 en 3.4.2 en Bijlage C
    - Export_APEX ..... Secties 3.4.4 en 3.4.5
    - Screenshots ..... Sectie 3.4.5
  - Documentatie_(latexcode_masterproef)

Bibliografie

[1] AnyChart, *Anychart, flash charting component , xml reference*, website `http://anychart.com/products/anychart/docs/xmlReference/index.html`.

[2] _____, *Why anychart?*, website `http://anychart.com/products/anychart/why_anychart/`.

[3] Yu-May Chang and Jeff Ullman, *Using oracle pl/sql*, website `http://infolab.stanford.edu/~ullman/fcdb/oracle/or-plsql.html`.

[4] Rus Heywood, *Uml use case diagrams: Tips and faq*, website `http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html`, 1999.

[5] Diana Lorentz, *Oracle® database sql reference,10g release 2 (10.2), part number b14200-02, create type*, online boek `http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/statements_8001.htm`, december 2005.

[6] _____, *Oracle® database sql reference,10g release 2 (10.2), part number b14200-02, sql functions*, online boek `http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/functions001.htm#i81407`, december 2005.

[7] Sheryl Maring, Rick Sapir, and Michael Wiesenberg, *Oracle® database java developer's guide, 10g release 1 (10.1), part number b12021-02, introduction to java in oracle database*, website `http://download.oracle.com/docs/cd/B14117_01/java.101/b12021/intro.htm`, juni 2004.

[8] Duncan Mein, *Duncan mein's blog, custom authentication / authorisation schemes (part 1)*, blog `http://djmein.blogspot.com/2007/07/custom-authentication-authorisation.html`, juli 2007.

[9] Oracle, *Oracle call interface (oci)*, website `http://www.oracle.com/technetwork/database/features/oci/index.html`.

[10] _____, *Sample charts, maps, gantts & trees, map with custom data point*, website `http://apex.oracle.com/pls/apex/f?p=36648:65:7243489697784145::NO:::`.

[11] plsql-tutorial.com, *Pl/sql tutorial, learn pl/sql in a simple way.*, website `http://www.plsql-tutorial.com/`.

[12] Patrick Sinke, *Custom authentication in apex (part 1)*, website `http://technology.amis.nl/blog/2623/custom-authentication-in-apex-part-1`, november 2007.

[13] Patrick Wolf, *Apex items and implicit type conversion*, website http://www.inside-oracle-apex.com/apex-items-and-implicit-type-conversion/, november 2006.

[14] _____, *Oracle apex 4.0: Cascading lovs/select lists*, website `http://www.inside-oracle-apex.com/oracle-apex-4-0-cascading-lovsselect-lists/`, februari 2010.