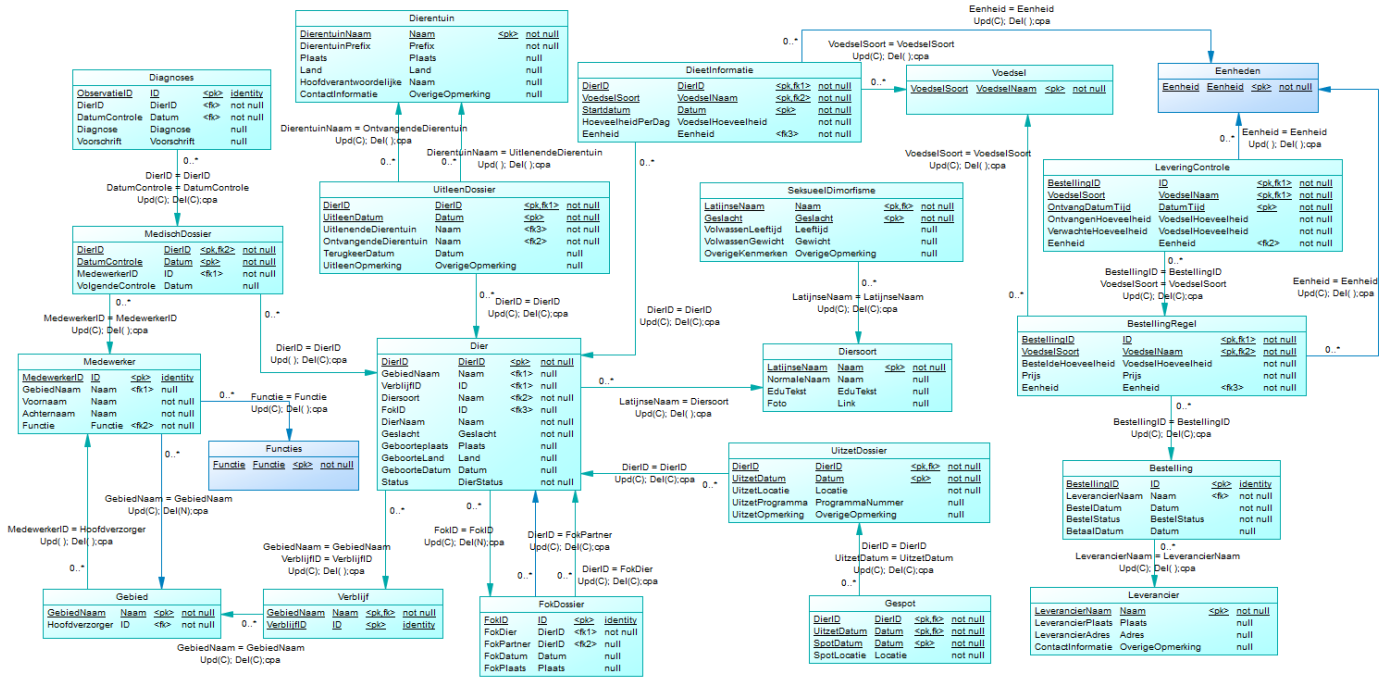
Technisch Ontwerp

Inleiding

In dit technisch ontwerp zullen we uitleggen hoe de functionaliteiten die benoemd zijn in het functioneel ontwerp gebouwd zullen gaan worden. Dit dient als een naslagwerk voor het ontwikkelteam en om eventuele designkwesties te verantwoorden.

Eerst zullen we het fysieke datamodel tonen die uit het conceptuele datamodel is gegenereerd samen met de beschrijvingen van de tabellen en kolommen die hierin gevonden kunnen worden. Verder worden ook de specificaties van de integriteitsregels die voortvloeien uit de constraints en business rules benoemd. Hierna zullen we de niet-functionele eisen van het systeem benoemen en een Definition of Done afspreken om de kwaliteit van de geschreven functionaliteiten te waarborgen. Aan het eind van het document worden de ontwerpbeslissingen met betrekking tot de implementatie van het project vastgelegd.

Fysiek Datamodel



Gemaakte keuzes

Medewerkers ID: We hebben besloten medewerkers een nummer te geven waardoor ze makkelijker uniek te identificeren zijn.

Gemiddelde volwassen leeftijd: Wij slaan de gemiddelde volwassen leeftijd en het gemiddelde gewicht op als varchar omdat er geen berekeningen op plaats hoeven te vinden en zodat een range aan waardes te definiëren is.

Beschrijving van tabellen en kolommen

In dit hoofdstuk zal in detail per onderdeel van het PDM de tabellen en de bijbehoorende kolommen beschreven worden. Hiermee hopen we te bereiken dat het duidelijk en concreet wordt. De lijst van tabellen kan gerefereerd naar worden indien verduidelijking nodig is.

Medewerker Deze tabel houdt de gegevens van alle medewerkers van het dierenpark bij. MedewerkerID: Om elke medewerker uniek te identificeren krijgen deze een ID mee. GebiedNaam: In het geval van een verzorger is dit het gebied waar de verzorger meedraait. Voornaam/Achternaam: De naam van de medewerker. Functie: De functie in het dierenpark waarvoor deze medewerker is ingehuurd.

Functie In deze tabel worden alle mogelijke functies bijgehouden die een medewerker van het bedrijf kan bekleden. Functie: De naam van de functie.

Gebied Deze tabel houdt een gebied van het park bij. Gebiednaam: De unieke naam van het gebied. Meestal is deze overeenkomstig met het thema van dat gebied. Hoofdverzorger: De verzorger die verantwoordelijk is voor het beheren van dit gebied en de dieren die hier verblijven.

Verblijf Deze tabel houdt een verblijf bij waar nul of meer dieren leven. GebiedNaam: Elk verblijf bevind zich in een specifiek gebied. VerblijfID: Om elk verblijf uniek te identificeren krijgen ze een ID mee.

Dier Deze tabel houdt de informatie van een dier van het dierenpark bij. DierID: Elk dier krijgt een uniek ID, hiermee kan systematisch naar gerefereerd worden. Het ID voor het dierenpark heeft eerst 3 letters van het diersoort, dan een koppelteken en tot slot een nummer. Indien het dier uit een andere dierentuin komt wordt deze opgeslagen met "EXT-" gevolgd door het ID dat is meegegeven vanuit die dierentuin. GebiedNaam: Het gebied waar dit dier verblijft. VerblijfID: Het verblijf waar dit dier leeft. FokID: ID van de foksessie waar het dier waarschijnlijk verwekt is. Diersoort: De latijnse naam van de diersoort waar dit dier onder valt. Diernaam: De roepnaam van dit dier. Geslacht: Dit veld houdt bij of het dier een mannetje, vrouwtje of van onbekend geslacht is. Geboorteplaats: Indien bekend, de stad waar dit dier geboren is. Geboorteland: Indien bekend, het land waar dit dier geboren is. Geboortedatum: Indien bekend, de datum waarop dit dier geboren is. Status: Een unieke status waar dit dier onder valt indien dit het geval is. (bijvoorbeeld: "Afwezig"/"Overleden")

Diersoort Deze tabel houdt de informatie bij van diersoorten. LatijnseNaam: De latijnse naam van het diersoort NormaleNaam: De nederlandse naam van het diersoort. EduTekst: Een stuk tekst dat het diersoort beschrijft. Dit bevat vaak de leefstijl van het dier en overige informatie die educatief is. Foto: Een link naar een foto van de diersoort.

SeksueelDimorfisme Deze tabel houdt de verschillende kenmerken van een diersoort bij per geslacht. LatijnseNaam: De latijnse naam van het diersoort waar dit dier onder valt. Geslacht: Welk geslacht van het huidige diersoort de gemmidelde informatie van wordt beschreven. VolwassenLeeftijd: De gemiddelde leeftijd voordat het dier als volwassen wordt beschouwd. VolwassenGewicht: Het gemiddelde gewicht van een volwassen dier. OverigeKenmerken: Een beschrijving van overige kenmerken van het diersoort.

MedischDossier Deze tabel houdt bij wanneer een dier een diagnose heeft gehad of nog moet hebben. DierID: Het unieke ID van het dier waar dit medisch dossier over gaat. DatumControle: De datum wanneer dit dier een medische diagnose heeft gehad. MedewerkerID: De dierenarts die deze medische controle heeft uitgevoerd. VolgendeControle: De datum waarop de volgende controle voor dit dier staat gepland.

Diagnose Deze tabel houdt bij wat de medische gebeurtenis voor een dier inhield op een specifieke datum. ObservatieID: elk voorschift heeft een uniek ID. DierID: Het unieke ID van het dier waar deze diagnose over gaat. DatumControle: de datum wanneer dit dier deze medische diagnose heeft gehad. Diagnose: Een beschrijving van een medische conditie die bij dit dier gevonden is. (bijvoorbeeld: "zwanger") Voorschrift: Een beschrijving/naam van een medicijn dat dit dier is voorgeschreven.

DieetInformatie Deze tabel houdt bij wat het dieet van een dier is. DierID: Het unieke ID van het dier waar dit dieet over gaat. VoedselSoort: De naam van het voedsel. Startdatum: De datum waarop dit dieet bij het dier voor het eerst is voorgeschreven. Als twee voedselsoorten dezelfde startdatum hebben bij eenzelfde dier zijn ze allebei deel van het dieet. HoeveelheidPerDag: De dagelijkse voedselhoeveelheid in het dieet van het dier. Eenheid: De eenheid van de kolom "HoeveelheidPerDag" (bijvoorbeeld: "Liter", "Kilogram")

Voedsel Deze tabel houdt bij welke soorten voedsel er zijn. VoedselSoort: De naam van het voedsel.

UitzetDossier Deze tabel houdt bij wat de details zijn bij het uitzetten van een dier. Deze tabel zal dus alleen voor een dier gevuld zijn als het dier is vrij gezet en zich niet meer in de dierentuin bevindt. DierID: Het unieke ID van het dier waar deze uitzetting over gaat. UitzetDatum: De datum waarop het dier is uitgezet. UitzetLocatie: De locatie waar het dier is uitgezet. UitzetProgramma: Het programma waar deze uitzetting deel van was. UitzetOpmerking: Een beschrijving of opmerking over deze uitzetting.

Gespot Deze tabel houdt bij de details van een spotting van een uitgezet dier. DierID: Het unieke ID van het dier waar deze spotting over gaat. UitzetDatum: De datum waarop dit dier is uitgezet. SpotDatum: De datum waarop deze spotting heeft plaats gevonden. SpotLocatie: De locatie waar het dier is gespot.

FokDossier Deze tabel houdt bij wanneer er tussen twee dieren een kind wordt verwekt. FokID: Elke Fok/paarsessie heeft een uniek ID. FokDier: Het DierID van de moeder van het verwekte kind. FokDatum: De datum waarop de verwekking plaatsvond. Wordt zo nauwkeurig mogelijk ingevuld, maar is niet altijd bekend. FokPartner: Het DierID van de vader van het verwekte kind. FokPlaats: De locatie waar het kind is verwekt.

UitleenDossier Deze tabel houdt bij wat de informatie is over een uitlening van een dier tussen Somerleyton en een andere dierentuin. DierID: Het ID van het dier dat uitgeleend wordt. UitleenDatum: De datum waarop de uitleen begint. UitlenendeDierentuin: De dierentuin waar dit dier origineel vandaan komt. OntvangendeDierentuin: De dierentuin waaraan dit dier wordt uitgeleend. TerugkeerDatum: De datum waarop is afgesproken dat het uitgeleende dier terug naar de uitlenende dierentuin wordt gebracht. UitleenOpmerking: Een veld waar mogelijke opmerkingen beschreven kunnen worden.

Dierentuin Deze tabel houdt bij welke dierentuinen Somerleyton uitleeninteracties mee heeft (gehad). DierentuinNaam: De naam van de dierentuin. Plaats: De locatie van waar de dierentuin zich bevindt. Land: Het land van waar de dierentuin zich bevindt. Hoofdverantwoordelijke: De naam van de contactpersoon van de dierentuin. ContactInformatie: Een tekstveld waar mogelijke contactgegevens opgeslagen kunnen worden.

Leverancier Deze tabel houdt bij welke leveranciers bekend zijn bij Somerleyton en de gegevens van deze leveranciers. LeverancierNaam: De bedrijfsnaam van de leverancier. LeverancierPlaats: De locatie van het leveranciersbedrijf. LeverancierAdres: Het adres van het leveranciersbedrijf. ContactInformatie: Een tekstveld waar

mogelijke contactgegevens opgeslagen kunnen worden.

Bestelling Deze tabel houdt bij wat de informatie van een bestelling richting een leverancier inhoud. BestellingID: Het unieke ID van een bestelling. LeverancierNaam: De bedrijfsnaam van de leverancier waarbij deze bestelling is geplaatst. BestelDatum: De datum waarop deze bestelling is aangemaakt. BestelStatus: De huidige status van de bestelling, dit beschrijft of de bestelling is opgesteld, wacht op betaling of al betaald is. BetaalDatum: De datum waarop deze bestelling is betaald.

BestellingRegel Deze tabel houdt de bestelingregels bij van de bestellingen. BestellingID: Het unieke ID van deze bestellingregel. VoedselSoort: De naam van het voedsel wat besteld wordt. BesteldeHoeveelheid: De hoeveelheid die van deze voedselsoort besteld wordt. Prijs: De afgesproken kost per eenheid van deze voedselsoort. Eenheid: De bijbehoorende eenheid van "BesteldeHoeveelheid".

LeveringControle In deze tabel wordt informatie over de leveringen die bij een bestelling horen vastgelegd. Zo is het mogelijk fouten en verschillen op te sporen. BestellingID: Het unieke ID van deze levering. VoedselSoort: De naam van het voedsel dat afgeleverd is. OntvangDatumTijd: De datum en het tijdstip van wanneer deze levering is ontvangen. OntvangenHoeveelheid: De hoeveelheid die van deze voedselsoort is afgeleverd. VerwachteHoeveelheid: De hoeveelheid die van deze voedselsoort geleverd zou moeten worden. Eenheid: De bijbehoorende eenheid van de OntvangenHoeveelheid en VerwachteHoeveelheid.

Eenheid In deze domeintabel worden alle gebruikte eenheden bijgehouden. Eenheid: De nam van de eenheid.

Use Cases

Hier wordt beschreven hoe elke use case geimplementeerd is. Alle use cases worden behandeld in de vorm van stored procedures die duidelijke benamingen hebben.

Use cases	Stored Procedure	beschrijving
UC 1.1	STP_InsertDier	nieuw dier toevoegen
UC 1.2	STP_InsertGebied	nieuw gebied toevoegen
UC 1.3	STP_InsertVerblijf	nieuw verblijf toevoegen
UC 1.4	STP_InsertDiersoort, STP_InsertSeksueeldimorfisme	nieuwe diereninformatie toevoegen
UC 1.5	STP_InsertFokdossier	nieuw fokdossier toevoegen
UC 1.6	STP_InsertUitleenDossier, STP_InsertDierentuin	nieuw uitleen dossier toevoegen
UC 1.7	STP_InsertUitzetDossier	nieuw uitzet dossier toevoegen
UC 1.8	STP_UpdateDier, STP_DeleteDier	dier aanpassen
UC 1.9	STP_UpdateGebied, STP_DeleteGebied	gebied aanpassen
UC 1.10	STP_UpdateVerblijf,STP_DeleteVerblijf	verblijf aanpassen
UC 1.11	STP_UpdateDiersoort, STP_DeleteVerblijf	diereninformatie aanpassen
UC 1.12	STP_UpdateFokdossier, STP_DeleteFokDossier	fokdossier aanpassen
UC 1.13	STP_UpdateUitleenDossier, STP_DeleteUitleendossier	uitleendossier aanpassen
UC 1.14	STP_UpdateUitzetDossier, STP_DeleteUitzetdossier	uitzetdossier aanpassen
UC 1.15	STP_OphalenDiereninformatie	uitdraaisel maken van diereninformatie
UC 1.16	STP_SelectDier	data ophalen van dier
UC 1.17	STP_SelectGebied	data ophalen van gebied
UC 1.18	STP_SelectVerblijf	data ophalen van verblijf
UC 1.19	STP_SelectDiersoort	data ophalen van diersoorten
UC 1.20	STP_SelectFokDossier	data ophalen van fokinformatie
UC 1.21	STP_SelectUitleenDossier	data ophalen van uitleendossier
UC 1.22	STP_SelectUitzetDossier	data ophalen van uitzetdossier
UC 1.23	STP_InsertGespot	nieuwe spotting toevoegen
UC 1.24	STP_UpdateGespot, STP_DeleteGespot	spotting aanpassen
UC 1.25	STP_SelectGespot	data ophalen van spotting
UC 1.26	STP_InsertMedischDossier	medisch dossier toevoegen
UC 1.27	STP_InsertDiagnoses, STP_UpdateMedischDossier, STP_DeleteMedischDossier	medisch dossier aanpassen
UC 1.28	STP_SelectMedischDossier	data ophalen van medisch dossier
UC 1.29	STP_InsertDieetInformatie	nieuwe voedingsinformatie toevoegen
UC 1.30	STP_InsertOntvangenGoederen	ontvangen goederen toevoegen
UC 1.31	STP_UpdateDieetinformatie, STP_DeleteDieetinformatie	voedingsinformatie aanpassen
UC 1.32	STP_SelectDieetinformatie	data ophalen van voedingsinformatie

Use cases	Stored Procedure	beschrijving
UC 1.33	STP_UpdateOntvangenGoederen, STP_DeleteOntvangenGoederen	ontvangen goederen aanpassen
UC 1.34	STP_SelectOntvangenGoederen	data ophalen van ontvangen goederen
UC 1.35	STP_InsertBestelling	nieuwe bestelling toevoegen
UC 1.36	STP_UpdateStatusBestelling	bestelling als betaald zetten
UC 1.37	STP_UpdateBestelling, STP_DeleteBestelling	bestelling aanpassen
UC 1.38	STP_UpdateBestellingRegel, STP_DeleteBestellingRegel	bestellingvoedsel aanpassen
UC 1.39	STP_SelectBestelling	data ophalen van bestelling
UC 1.40	STP_SelectBestellingStatus	ophalen status van bestelling
UC 1.41	STP_UpdateHoofdverzorger	de hoofdverzorger van een gebied aanpassen
UC 1.42	STP_UpdateGebiedVerzorger	het gebied van een verzorger aanpassen
UC 1.43	STP_SelectMedewerker	data ophalen van een medewerker
UC 1.44	STP_InsertLeverancier	nieuwe leverancier toevoegen
UC 1.45	STP_UpdateLeveranciers, STP_DeleteLeverancier	leverancier aanpassen
UC 1.46	STP_SelectLeverancier	data ophalen van leverancier
UC 1.47	STP_SelectVoedsel	data ophalen over voedselsoorten
UC 1.48	STP_InsertVoedsel	nieuw voedselsoort toevoegen
UC 1.49	STP_SelectFuncties	data ophalen van functies
UC 1.50	STP_InsertFuncties	nieuwe functie toevoegen
UC 1.51	STP_UpdateVoedselsoort, STP_DeleteVoedsel	voedselsoorten aanpassen
UC 1.52	STP_UpdateFuncties, STP_DeleteFunctie	functies aanpassen
UC 1.53	STP_InsertMedewerker	nieuwe medewerker toevoegen

Integriteitsregels

Hieronder worden de integriteitsregels met ieder een eigen nummer beschreven. De integriteitsregels komen overeen met de constraints en business rules die beschreven zijn in het funtioneel ontwerp. Per integriteitsregel wordt een specificatie gegeven en wordt beschreven hoe ze worden geïmplementeerd in de database.

IR1 komt overeen met C1 en BR 1 LeverdatumBestelling

- Specificatie: Voor iedere bestelling mag de ontvangdatum niet voor de besteldatum liggen.
- Implementatie: Trigger TRG_LeverdatumBestelling op de tabel LeveringControle.
- After: Insert, Update

IR2 komt overeen met C2 en BR 2 GeboortedatumDier

- Specificatie: De geboortedatum van een dier kan niet eerder zijn dan de geboortedatum van zijn ouders.
- Implementatie: Trigger TRG_GeboortedatumDier op de tabel Dier.
- After: Insert, Update

IR3 komt overeen met C3 en BR 3 VoederDatum

- Specificatie: Een dier kan alleen gevoerd worden als deze is geboren. Dit houdt in dat de startdatum niet voor de geboortedatum kan liggen.
- Implementatie: Trigger TRG_VoederDatum op de tabel DieetInformatie.
- After: Insert, Update

IR4 komt overeen met C4 en BR 4 MedischDossierDatum

- Specificatie: Een medisch dossier kan alleen worden opgesteld voor een dier dat is geboren. De datum van het medischdossier kan niet voor de geboortedatum zijn.
- Implementatie: Trigger TRG_MedischDossierDatum op de tabel MedischDossier.
- After: Insert, Update

IR5 komt overeen met C5 en BR 5 FokgeschiedenisDatum

- Specificatie: Er kan alleen met een dier gefokt worden als deze ook is geboren. Dit houdt in dat de fokdatum na de geboortedatum moet liggen.
- Implementatie: Trigger TRG_FokgeschiedenisDatum op de tabel FokDossier.
- After: Insert, Update

IR6 komt overeen met C6 en BR 6 TerugkeerDatum

- Specificatie: De terugkeerdatum van een dier dat uitgeleend wordt moet na de uitleendatum van een dier zijn.
- Implementatie: Check-constraint CHK_TerugkeerDatum op de tabel UitleenDossier.

IR7 komt overeen met C7 en BR 7 GeslachtFokDier

- Specificatie: Als een dier wordt gebruikt om mee te fokken kan dit niet met een dier van hetzelfde geslacht zijn als beide geslachten bekend zijn.
- Implementatie: Trigger TRG_GeslachtFokDier op de tabel FokDossier.
- After: Insert, Update

IR8 komt overeen met C8 en BR 8 UitleenDatum

- Specificatie: Een dier kan alleen worden uitgeleend als deze ook geboren is. De uitleendatum moet na de geboortedatum liggen.
- Implementatie: Trigger TRG_UitleenDatum op de tabel UitleenDossier.
- After: Insert, Update

IR9 komt overeen met C9 en BR 9 UitzetDatum

- Specificatie: Een dier kan alleen uitgezet worden als deze is geboren. De uitzetdatum moet na de geboortedatum zijn.
- Implementatie: Trigger TRG_UitzetDatum op de tabel UitzetDossier.
- After: Insert, Update

IR10 komt overeen met C10 en BR 10 GeslachtMoeder

- Specificatie: De moeder(FokDier) van een dier moet het geslacht van een vrouwtje(F) hebben.
- Implementatie: Trigger TRG_GeslachtMoeder op de tabel FokDossier.
- After: Insert, Update

IR11 komt overeen met C11 en BR 11 GeslachtVader

- Specificatie: De vader(FokPartner) van een dier moet het geslacht van een mannetje(M) hebben.
- Implementatie: Trigger TRG_GeslachtVader op de tabel FokDossier.
- After: Insert, Update

IR12 komt overeen met C12 en BR 12 FokLocatie

- Specificatie: Een dier dat in FokDossier staat moet dezelfde fokplaats hebben als de locatie waar het dier op dat moment verblijft.
- Implementatie: Trigger TRG_FokLocatie op de tabel FokDossier.
- After: Insert, Update

IR13 komt overeen met C13 en BR 13 BestellingBetalen

- Specificatie: De Betaaldatum kan alleen worden ingevuld op het moment dat de bestelstatus op 'Betaling_Nodig' staat.
- Implementatie: Check-constraint CHK_BestellingBetalen op de tabel Bestelling.

IR14 komt overeen met C14 en BR 14 BetaalDatumBestelling

- Specificatie: De Betaaldatum in Bestelling moet gelijk zijn aan of later zijn dan de BestelDatum.
- Implementatie: Check-constraint CHK_BetaalDatumBestelling op de tabel Bestelling.

IR15 komt overeen met C15 en BR 15 DatumDierGespot

- Specificatie: De SpotDatum in Gespot moet na of gelijk zijn aan de UitzetDatum in Uitzetdossier liggen.
- Implementatie: Check-constraint CHK_DatumDierGespot op de tabel Gespot.

IR16 komt overeen met C16 en BR 16 DiagnoseDoorDierenarts

- Specificatie: De medewerkerID in MedischDossier moet de functie dierenarts hebben in medewerker.
- Implementatie: Trigger TRG_DiagnoseDoorDierenarts op de tabel MedischDossier.
- After: Insert, Update

IR17 komt overeen met C17 en BR 17 UitgeleendDier

- Specificatie: Een dier in UitleenDossier kan niet meerdere registraties hebben tussen de UitleenDatum en TerugkeerDatum. De uitleenDatum van een andere registratie van hetzelfde dier kan dus niet in de periode liggen van de uitleen en terugkeerdatum.
- Implementatie: Trigger TRG_UitgeleendDier op de tabel UitleenDossier.
- After: Insert, Update

IR18 komt overeen met C18 en BR 18 DierInHetWild

- Specificatie: Een dier kan niet voorkomen in UitzetDossier als deze in Dier een VerblijfID heeft.
- Implementatie: Trigger TRG_DierInHetWild op de tabel UitzetDossier.
- After: Insert, Update

IR19 komt overeen met C19 en BR 19 ZwangerschapDier

- Specificatie: Alleen een dier met geslacht vrouwtje kan met een zwangerschap gediagnosticeerd worden.
- Implementatie: Trigger TRG_ZwangerschapDier op de tabel Diagnose.
- After: Insert, Update

IR20 komt overeen met C20 en BR 20 GeboorteDatumInToekomst

- Specificatie: Op het moment dat een nieuw dier wordt toegevoegd aan Dier mag de geboortedatum niet in de toekomst liggen.
- Implementatie: Check-constraint CHK_GeboorteDatumInToekomst op de tabel Dier.

IR21 komt overeen met C21 en BR 21 FokkenMetFamilie

- Specificatie: Een dier kan niet fokken met een ander dier dat dezelfde vader en/ of moeder heeft. Een dier kan ook niet direct met een van de ouders fokken.
- Implementatie: Trigger TRG_FokkenMetFamilie op de tabel FokDossier.
- After: Insert, Update

IR22 komt overeen met C22 en BR 22 HoofdverzorgerGebied

- Specificatie: Een hoofdverzorger in een gebied kan alleen hoofdverzorger zijn als deze ook de functie van hoofdverzorger heeft.
- Implementatie: Trigger TRG_HoofdverzorgerGebied op de tabel Gebied.
- After: Insert, Update

IR23 komt overeen met C23 en BR 23 UitleenDierentuin

- Specificatie: Een dier dat uitgeleend wordt heeft als uitlenende of als ontvangende dierentuin altijd Somerleyton Animal Park.
- Implementatie: Check-constraint CHK_UitleenDierentuin op de tabel UitleenDossier.

IR24 komt overeen met C24 en BR 24 MedischDossierUitgezetDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in MedischDossier na de UitzetDatum als deze voorkomt in UitzetDossier.
- Implementatie: Trigger TRG_MedischDossierUitgezetDier op de tabel MedischDossier.
- After: Insert, Update

IR25 komt overeen met C25 en BR 25 FokDossierUitgezetDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in FokDossier na de UitzetDatum als deze voorkomt in UitzetDossier.
- Implementatie: Trigger TRG_FokDossierUitgezetDier op de tabel FokDossier.
- After: Insert, Update

IR26 komt overeen met C26 en BR 26 UitleenDossierUitgezetDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in UitleenDossier na de UitzetDatum als deze voorkomt in UitzetDossier.
- Implementatie: Trigger TRG_UitleenDossierUitgezetDier op de tabel UitleenDossier.
- After: Insert, Update

IR27 komt overeen met C27 en BR 27 VoedselInfoUitgezetDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in DieetInformatie na de UitzetDatum als deze voorkomt in UitzetDossier.
- Implementatie: Trigger TRG_VoedselInfoUitgezetDier op de tabel DieetInformatie.
- After: Insert, Update

IR28 komt overeen met C28 en BR 28 VerblijfUitgezetDier

- Specificatie: Een dier dat voorkomt in UitzetDossier heeft geen verblijfID in Dier.
- Implementatie: Trigger TRG_VerblijfUitgezetDier op de tabel UitzetDossier.
- After: Insert, Update

IR29 komt overeen met C29 en BR 29 MedischDossierDoodDier

- Specificatie: Een dier dat is overleden heeft geen nieuwe registraties in MedischDossier.
- Implementatie: Trigger TRG_MedischDossierDoodDier op de tabel MedischDossier.
- After: Insert, Update

IR30 komt overeen met C30 en BR 30 FokDossierDoodDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in Fokdossier als het dier de status 'Overleden' heeft.
- Implementatie: Trigger TRG_FokDossierDoodDier op de tabel FokDossier.
- After: Insert, Update

IR31 komt overeen met C31 en BR 31 UitleenDossierDoodDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in UitleenDossier als het dier de status 'Overleden' heeft.
- Implementatie: Trigger TRG_UitleenDossierDoodDier op de tabel UitleenDossier.
- After: Insert, Update

IR32 komt overeen met C32 en BR 32 VoedselInfoDoodDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in DieetInformatie als het dier de status 'Overleden' heeft.
- Implementatie: Trigger TRG_VoedselInfoDoodDier op de tabel DieetInformatie.
- After: Insert, Update

IR33 komt overeen met C33 en BR 33 VerblijfDoodDier

- Specificatie: Een dier kan geen VerblijfID meer hebben in Dier als deze de status overleden heeft.
- Implementatie: Check-constraint CHK_VerblijfDoodDier op de tabel Dier.

IR34 komt overeen met C34 en BR 34 UitzettenDoodDier

- Specificatie: Een dier kan geen nieuwe registratie krijgen in UitzetDossier als het dier de status 'Overleden' heeft.
- Implementatie: Trigger TRG_UitzettenDoodDier op de tabel UitzetDossier.

- After: Insert, Update

IR35 komt overeen met C35 en BR 35 VerzorgerInGebied

- Specificatie: Alleen een medewerker met de functie verzorger/ hoofdverzorger heeft een gebiednaam in de tabel Medewerker.
- Implementatie: Check-constraint CHK_VerzorgerInGebied op de tabel Medewerker.

Architecturaal prototype

Met een architecturaal prototype wordt gekeken of de wensen van de opdrachtgever van Somerleyton Animal park technisch mogelijk zijn. De opdrachtgever wilt graag 2 staging area's hebben naast de relationele database.

Met de eerste staging area moet het mogelijk zijn dat mensen van buiten de dierentuin informatie kunnen ophalen over de verschillende diersoorten. Het is de bedoeling dat deze informatie 1 keer per dag automatisch van de relationele database overgezet wordt naar de staging area.Met de tweede staging area moet de dierentuin informatie kunnen uitwisselen met andere dierentuinen over bijvoorbeeld het fokdossier van dieren. Andere dierentuinen moeten informatie in deze staging area kunnen zetten en ophalen. De gegevens die in deze staging area staan moeten uiteindelijk ook terecht komen in de relationele database van Somerleyton Animal Park.

Als eerst is gekeken of het automatisch heen en weer gaan van gegevens mogelijk is tussen twee verschillende relationele databases. Er zijn twee databases opgezet met beide exact dezelfde tabellen waarbij 1 database gevuld is met gegevens en de ander leeg is. Vervolgens is een zogenaamde 'job' gemaakt die iedere minuut(dit kan elke gewenste tijd zijn) gegevens vanuit de ene database automatisch overzet naar de andere database. De gegevens gingen daadwerkelijk iedere minuut automatisch van de ene naar de andere database. De staging area in de vorm van een relationele database is dus vrij eenvoudig te realiseren. De scripts voor dit prototype zijn hier te vinden.

Na gesprekken te hebben gevoerd met de opdrachtgever bleek echter dat hij liever NoSQL databases heeft als staging area. In het volgende hoofdstuk wordt hier verder op in gegaan.

MongoDB

Voor de staging area's hebben we ervoor gekozen om MongoDB te gebruiken, dat is een database met veel ondersteuning en is ideaal voor deze situatie. Om dit te testen hebben we via MongoDB atlas een kleine server online gezet waar wat test data opgezet is. Hierdoor hebben we een beetje kennis kunnen maken met MQL de taal van mongodb. Hierdoor weten we dat het mogelijk is om op deze manier de staging area's toe te passen. Wat nog wel onderzocht moet worden is de communicatie tussen onze SQL server en deze staging area's. We hebben hiervoor commerciele oplossingen voor gevonden maar die zijn veel te duur om voo dit project te gebruiken. Daarom zullen we dit zelf moeten ontwikkelen. De educatieve staging area hoeft alleen data te ontvangen en dat is goed te doen, omdat mongodb net als SQL statement communicatie accepteert. Voor de andere staging area moet mongodb in een ideale situatie ook terug communiceren. Hoe dat werkt moeten we nog onderzoeken. Dit archtecturele prototype is bereikbaar op: `mongodb+srv://ISEC2:k5LoS2GUPz6F@cluster0.b2pzk.mongodb.net/test?authSource=admin&replicaSet=atlas-m3icjf-shard-0&readPreference=primary&appname=mongodb-vscode%200.7.0&ssl=true`

Niet-functionele eisen

Niet-functionele eisen beschrijven hoe je project zich zou moeten gedragen. Deze requirements passen zich aan naarmate de benodigdheden van het project zich veranderen. De niet-functionele eisen zijn beschreven zoals voorgeschreven in Vironit,2021.

Niet-Functionele eis	Beschrijving
NFR 1.1	Op de database moeten 10 mensen tegelijkertijd een stored procedure kunnen uitvoeren die binnen 2 seconden resultaat geeft.
NFR 1.2	De database moet via stored procedures kunnen communiceren met de staging area's.
NFR 1.3	De database moet via stored procedures de opgeslagen informatie kunnen aanpassen.
NFR 1.4	De database moet via stored procedures informatie kunnen toevoegen.
NFR 1.5	De database moet via stored procedures de opgeslagen informatie kunnen verwijderen.
NFR 1.6	De database moet via stored procedures de opgeslagen informatie kunnen versturen.
NFR 1.7	De database moet 95% van de tSQLt tests succesvol afronden.
NFR 1.8	De database moet alleen bereikbaar zijn voor rollen met toegang.
NFR 1.9	De database is gemaakt met MSSQL.
NFR 1.10	De database is getest met tSQLt.
NFR 1.11	Een gebruiker kan alleen gebruik maken van stored procedures waar hij rechten voor heeft.
NFR 1.12	Een gebruiker heeft geen directe toegang tot tabellen.
NFR 1.13	De database moet 24/7 beschikbaar zijn voor gebruikers.

CRUD matrix

Elke Use Case heeft bepaalde rechten nodig voor de bijbehoorende entiteiten in de database. In de CRUD matrix tabel hieronder is weergegeven voor elke Use Case welke rechten op welke entiteiten zij nodig hebben om succesvol uitgevoerd te kunnen worden. Met het opstellen van deze tabel creëren we een overzicht van use case rechten. Dit kan gebruikt worden bij het opstellen van de stored procedures zodat het duidelijk is met welke entiteiten deze stored procedure te maken heeft maar ook welke rechten deze nodig heeft.

Verder kan het gebruikt worden om te zien welke entiteiten veel gebruikt worden en op welke manier, deze informatie is nuttig bij het schrijven van indexes. Door indexes op de juiste manier op te stellen kan het gebruik van deze entiteiten versnellen. Door de rechten te linken aan de use cases wordt het lastig om per ongeluk foute informatie in de database toe te voegen.

Legenda: C = Create R = Read U = Update D = Delete

CRUD	Dier	Gebied	Verblijf	Diersoort	SeksueelDimorfisme	FokDossier	UitleenDossier	Dierentuin	UitzetDossier	Gespot	MedischDossie
UC 1.1	C										
UC 1.2		C									
UC 1.3			C								
UC 1.4				C	C						
UC 1.5						C					
UC 1.6							C	C,R			
UC 1.7									C		
UC 1.8	U,D										
UC 1.9		U,D									
UC 1.10			U,D								
UC 1.11				U,D	U,D						
UC 1.12						U,D					
UC 1.13							U,D	C,R,U,D			
UC 1.14									U,D	R	
UC 1.15	R			R	R						
UC 1.16	R										
UC 1.17		R									
UC 1.18			R								
UC 1.19				R	R						
UC 1.20						R					
UC 1.21							R	R			
UC 1.22									R	R	
UC 1.23										C	
UC 1.24										U,D	
UC 1.25										R	

CRUD	Dier	Gebied	Verblijf	Diersoort	SeksueelDimorfisme	FokDossier	UitleenDossier	Dierentuin	UitzetDossier	Gespot	MedischDossie
UC 1.26											C
UC 1.27											U,D
UC 1.28											R
UC 1.29											
UC 1.30											
UC 1.31											
UC 1.32											
UC 1.33											
UC 1.34											
UC 1.35											
UC 1.36											
UC 1.37											
UC 1.38											
UC 1.39											
UC 1.40											
UC 1.41		U,D									
UC 1.42		R									
UC 1.43											
UC 1.44											
UC 1.45											
UC 1.46											
UC 1.53											

Om de tabel niet groter te maken zijn de laatste 4 usecases erbuiten gelaten. UC 1.47 READ op voedselsoorten UC 1.48 INSERT op voedselsoorten UC 1.49 READ op functies UC 1.50 INSERT op functies UC 1.51 UPDATE, DELETE op voedselsoorten UC 1.52 UPDATE, DELETE op functies

Indexing

Relationele Database

We zijn bij de stored procedures langs gegaan en hebben tijdens het uitvoeren hiervan gekeken hoe de database de queries doorloopt. Dit deden we om mogelijke verbeteringen te kunnen spotten bij deze procedures. Indien mogelijk wilde we dingen zoals onnodige table scans vermijden. Het bleek echter dat met de opstelling die we hebben gemaakt dat de database al vlotte methodes toe past om de queries uit te voeren. Ook bij de meer complexe stored procedures kwam er geen probleem naar voren. Indien we een probleem met de methodes zouden vinden kunnen deze verbeterd worden door het in te voegen van indexes, maar dit bleek dus niet nodig uit de tests die we gedaan hebben.

Staging Area

Onze staging area's zijn gemaakt in mongoDB een database vooral ontwikkeld is met het oog op verticale en horizontale data opslag. Daarom zijn Mongo databases vaak al heel geoptimaliseerd.

In de Educatieve staging area is maar 1 tabel aangemaakt, daarom zijn er bij de educatieve staging area geen indexes van toepassing.

In de Fokdossier staging area is ook maar 1 tabel aangemaakt, daarom zijn er bij de fokdossier staging area geen indexes van toepassing. Bij deze staging area kan het in de toekomst wel nodig zijn om indexes toe te voegen omdat deze nog verder uitgebreid zou kunnen worden met alle dieren en ook alle uitleendossiers. Dan zou bijvoorbeeld een multikey index op alle dossiers van 1 dier interessant kunnen zijn. maar dat is nu nog niet van toepassing.

Concurrency

Bij het gebruik van een database is de concurrency en correctheid van de gegevens erg belangrijk, ook wanneer er meerdere mensen tegelijk aanpassingen willen maken. Daarom zullen we hier aandacht geven aan het toevoegen van transaction isolation levels aan de create en update use cases. In deze stored procedures kunnen concurrency issues ontstaan wanneer er een bepaalde constraint wordt gecheckt voordat een aanpassing of toevoeging gedaan wordt. Als tussen het checken en aanpassen door een andere gebruiker een aanpassing gedaan wordt kan het zijn dat de constraint omzeild wordt. Door de isolation levels aan te passen kunnen bepaalde records vergrendeld worden voor andere gebruikers totdat de aanpassingen gedaan zijn, waardoor de concurrency problemen vermeden kunnen worden. Hieronder volgt eerst een analyse van de use cases waar dit van toepassing is.

Insert Use Cases

Geen concurrency issues: UC 1.1, 1.3, 1.4, 1.23, 1.35, 1.45, 1.48, 1.50, 1.53, 1.6, 1.4, 1.27

UC 1.2: Checkt of medewerker hoofdverzorger is of dat medewerker al een hoofdverzorger voor een ander gebied is voordat deze aan een gebied wordt toegevoegd. UC 1.5: Dier wordt gecheckt voordat hij wordt toegevoegd aan fokdossier. UC 1.6: Dier wordt gecheckt voordat hij wordt toegevoegd aan uitleendossier. UC 1.26: Dier en medewerker worden gecheckt voordat ze worden toegevoegd aan medischdossier. UC 1.29: Dier wordt gecheckt voordat hij wordt toegevoegd aan dieetinformatie. UC 1.30: De ontvangdatum van een levering wordt vergeleken met de besteldatum en er wordt gecheckt dat de bestelling niet al betaald is. UC 1.35 (Bestellingregel): Checkt of Bestelling al betaald is. UC 1.7: Checkt of dier al overleden is of dat de uitzetdatum na geboortedatum ligt voordat deze aan uitzetdossier wordt toegevoegd.

Update Use Cases

Geen concurrency issues: UC 1.8, 1.9, 1.10, 1.11, 1.12, 1.24, 1.31, 1.27, 1.33, 1.37, 1.38, 1.51, 1.52

UC 1.13: Dier wordt gecheckt voordat hij wordt geupdate in uitleendossier. UC 1.14: Checkt of dier al overleden is of dat de uitzetdatum na geboortedatum ligt voordat deze in uitzetdossier wordt geupdate. UC 1.33: De ontvangdatum van een levering wordt vergeleken met de besteldatum en er wordt gecheckt dat de bestelling niet al betaald is. UC 1.36: Bestelstatus wordt gecheckt voordat hij wordt aangepast. UC 1.41: Checkt of hoofdverzorger al bestaat in gebied. UC 1.42: Checkt of medewerker wel een verzorger is.

Simulaties isolation levels

Om te begrijpen welke isolation level nodig is voor elke stored procedure simuleren we een situatie waar meerdere gebruikers de use case uitvoeren. In de meeste gevallen zullen twee gebruikers dezelfde constraint na elkaar checken, concluderen dat hij niet overtreden wordt door de uit te voeren insert of update en vervolgens beiden de query uitvoren terwijl deze maar een keer uitgevoerd mag worden. Hieronder staat een voorbeeld van zo'n analyse voor Insert Use Case 1.2, waar twee gebieden toegevoegd proberen te worden met dezelfde hoofdverzorger.

t=	**Transaction T1** **Read Committed**	**Transaction T2** **Read Committed**
1	EXEC STP_Insert @Gebiednaam = 'Savanne', @Hoofdverzorger = 1 (IN DE STORED PROCEDURE) IF EXISTS (SELECT 1 FROM MEDEWERKER WHERE FUNCTIE = 'Hoofdverzorger' AND MEDEWERKERID = @HOOFDVERZORGER AND GEBIEDNAAM IS NOT NULL) THROW 51042, 'Deze hoofdverzorger heeft al een gebied', 16; S-locks gevraagd en gegeven want er zijn nog geen locks. S-locks vervallen na het lezen.	
2		EXEC STP_Insert @Gebiednaam = 'Vlindertuin', @Hoofdverzorger = 1 (IN DE STORED PROCEDURE) IF EXISTS (SELECT 1 FROM MEDEWERKER WHERE FUNCTIE = 'Hoofdverzorger' AND MEDEWERKERID = @HOOFDVERZORGER AND GEBIEDNAAM IS NOT NULL) THROW 51042, 'Deze hoofdverzorger heeft al een gebied', 16; S-locks gevraagd en gegeven want er zijn nog geen locks. S-locks vervallen na het lezen.
3	INSERT INTO Gebied(GEBIEDNAAM,HOOFDVERZORGER) VALUES (@GebiedNaam,@HOOFDVERZORGER) X-lock gevraagd en gegeven want het is een nieuw record zonder locks. Deze lock wordt vastgehouden tot het eind van de transactie.	
4		INSERT INTO Gebied(GEBIEDNAAM,HOOFDVERZORGER) VALUES (@GebiedNaam,@HOOFDVERZORGER) X-lock gevraagd en gegeven want het is een nieuw record zonder locks. Deze lock wordt vastgehouden tot het eind van de transactie.
5	UPDATE MEDEWERKER SET GEBIEDNAAM = @GebiedNaam WHERE MEDEWERKERID = @HOOFDVERZORGER X-locks gevraagd en gegeven want op dit record staan nog geen locks. Deze lock wordt vastgehouden tot het eind van de transactie.	
6		UPDATE MEDEWERKER SET GEBIEDNAAM = @GebiedNaam WHERE MEDEWERKERID = @HOOFDVERZORGER X-locks gevraagd en niet gegeven want op dit record staan een X-lock van de andere gebruiker. De transactie wacht totdat hij door kan gaan.
7	COMMIT TRANSACTION; X-locks worden losgelaten	
8		COMMIT TRANSACTION; Sinds de locks van de andere transactie nu weg zijn kan deze transactie z'n X-locks ontvangen en de eerdere update query uitvoeren.

Aan het eind van deze situatie zijn er twee gebieden toegevoegd met dezelfde hoofdverzorger en heeft deze medewerker het gebied van transactie T2 als zijn gebied staan in de medewerker tabel. Om deze ongewenste situatie te voorkomen zetten we het transaction isolation level op Repeatable Read. Nu wordt dezelfde situatie nagespeeld met de strictere isolatie.

t=	**Transaction T1** **Repeatable Read**	**Transaction T2** **Repeatable Read**

1	`EXEC STP_Insert @Gebiednaam = 'Savanne', @Hoofdverzorger = 1`
	`(IN DE STORED PROCEDURE)`
	`IF EXISTS (`
	`SELECT 1 FROM MEDEWERKER`
	`WHERE FUNCTIE = 'Hoofdverzorger'`
	`AND MEDEWERKERID = @HOOFDVERZORGER`
	`AND GEBIEDNAAM IS NOT NULL`
	`)`
	`THROW 51042, 'Deze hoofdverzorger heeft al een gebied', 16;`
	
	S-locks gevraagd en gegeven want er zijn nog geen locks. S-locks worden vastgehouden.	

2		`EXEC STP_Insert @Gebiednaam = 'Vlindertuin', @Hoofdverzorger = 1`
		`(IN DE STORED PROCEDURE)`
		`IF EXISTS (`
		`SELECT 1 FROM MEDEWERKER`
		`WHERE FUNCTIE = 'Hoofdverzorger'`
		`AND MEDEWERKERID = @HOOFDVERZORGER`
		`AND GEBIEDNAAM IS NOT NULL`
		`)`
		`THROW 51042, 'Deze hoofdverzorger heeft al een gebied', 16;`
		
		S-locks gevraagd en niet gegeven want er staat een lock van transactie T1 op medewerker 1. Transactie wacht totdat locks weg zijn.

3	`INSERT INTO Gebied(GEBIEDNAAM,HOOFDVERZORGER)`
	`VALUES (@GebiedNaam,@HOOFDVERZORGER)`
	
	X-lock gevraagd en gegeven want het is een nieuw record zonder locks. Deze lock wordt vastgehouden tot het eind van de transactie.	

4	`UPDATE MEDEWERKER`
	`SET GEBIEDNAAM = @GebiedNaam`
	`WHERE MEDEWERKERID = @HOOFDVERZORGER`
	
	X-locks gevraagd en gegeven want dit record is al eerder gelocked door deze transactie. Deze lock wordt vastgehouden tot het eind van de transactie.	

5	`COMMIT TRANSACTION;` X-locks worden losgelaten	
6		Transactie kan weer doorgaan doordat locks van T1 losgelaten zijn. De query wordt uitgevoerd en er wordt een error gethrowd, sinds de andere transactie het gebied van de medewerker heeft geupdate.

Zoals je ziet wordt de insert van T2 geblokkeerd net zoals we willen. In de meeste gevallen voldoet een Repeatable Read voor het voorkomen van dit soort problemen. Het hoogste isolatielevel, Serializable, is alleen toepasselijk wanneer er een bereik aan waarden gecheckt en deze gelocked moet worden. Uit een snelle blik naar de eerder benoemde use cases blijkt dat Insert UC 1.6 en Update UC 1.13 BETWEEN gebruiken om te kijken of er al een uitleendossier bestaat die overlapt met degene die toegevoegd/geüpdate worden. Om te zorgen dat ook in deze gevallen de concurrency behouden wordt zullen we deze procedures Serializable maken en geven we de rest het isolatielevel Repeatable Read.

Definition of Done

Om te zorgen dat alle use cases en constraints zonder problemen geïmplementeerd worden volgens een vooraf afgesproken kwaliteitsstandaard zullen we hier een Definition of Done definiëren waar al onze functionaliteiten aan moeten voldoen. Deze definitie zal dienen als een checklist die het ontwikkelteam tijdens het proces kan gebruiken om de voortgang op een bepaalde functionaliteit te meten. Deze Definition of Done is als volgt:

- Unit tests zijn geschreven.
- Code is geschreven.
- Code is gedocumenteerd.
- Code is herleidbaar naar de documentatie.
- Code voldoet aan standardisatie (Captalisatie, Naamgeving, Commentaar, etc.)
- Code uitgerold op de gemeenschappelijke server.
- Code is transactie (stored procedure) en multi-input (triggers) bestendig.

- Unit en Functionele tests zijn geslaagd.

Staging Area's

User Roles

Voor de educatieve staging area is er eigenlijk maar 1 user role nodig namelijk die van gebruiker. Deze user role mag alleen data ophalen en niet verwijderen. Het verwijderen en toevoegen van data in deze staging area gebeurt automatisch. Buiten het gebruik van een user role moet iemand's IP-adres toegevoegd zijn aan de whitelist om gebruik te kunnen maken van de database.

Username: Gebruiker WW: SomerleytonGebruiker

Voor Somerleyton zelf is voor de Fokdossier Staging area een gebruiker aangemaakt die alles kan.

Username: ISEC2 WW: SomerleytonAdmin

Voor andere dierentuinen is een account aangemaakt dat fokdossiers kan invoegen en uitlezen. Maar niet verwijderen

Username: AndereDierentuin WW: DierentuinPassword

Testen

Het gebruik van tSQLt is helaas niet mogelijk in mongoDB, daarnaast zijn onze staging area's veel kleiner en minder gevoelig voor fouten. Daarom hebben we er voor gekozen om deze handmatig te testen. Het hoe en wat daarvan staat beschreven in /Scripts/Staging_Area_tests.bat

Handleiding

Voordat je gebruik kunt maken van mongoDB moet MongoSH geinstalleerd zijn.

De Staging area zet je op met bat files die allemaal command line commands uitvoeren. Deze zijn te vinden in het mapje Scripts. Deze is ook te vinden in een cloudcluster waar de staging area live op draait. (URI:"mongodb+srv://cluster0.b2pzk.mongodb.net/Somerleyton"). Voor het aanmaken van de database voer je de toepasselijke create scripts uit. Om de data over te zetten van SQL naar Mongo voer je MSSQL-MONGODB.bat uit. Daarna is de database gereed voor gebruik. Het overzetten van SQL naar mongo is helaas nog wel handmatig en zou eventueel in een volgende versie automatisch gemaakt kunnen worden.

Als laatste stap kun je dus connecten met je mongoDB met bijvoorbeeld:"mongosh "mongodb+srv://cluster0.b2pzk.mongodb.net/Somerleyton" --username ISEC2 --password k5LoS2GUPz6F

db.getCollection('diersoort').aggregate([{ $lookup: { from: 'seksueeldimorfisme', localField: 'LATIJNSENAAM', foreignField: 'LATIJNSENAAM', as: 'Diereninformatie' } }]);"

Importeren en exporteren van data

Voor het versturen van data van SQL naar MongoDB hebben we gebruik gemaakt van bcp, dat is een command line tool van windows zelf om data om te zetten vanaf een SQL server naar json of bijvoorbeeld excel. De procedure hiervan staat /Scripts/Educatieve%20Staging%20Area/MSSQL-MONGODB.bat. De data wordt omgezet naar JSON waarna het met de mongoimport command line tool wordt ingevoerd in de mongo database. Voor het importeren van data vanaf mongoDB naar SQL Server wordt MongoExport en bcp gebruikt. Dit gebeurt voor alsnog wel handmatig.

Ontwerpbeslissingen

In dit hoofdstuk leggen we de gemaakte keuzes vast met de redenen waarom we deze keuzes hebben gemaakt.

Beslissingen over Niet-functionele eisen

Niet functionele eisen zijn de eisen die vooral de betrouwbaarheid, veiligheid, traceerbaareid en bruikbaarheid van een product beschrijven. Omdat onze database niet op grote schaal gebruikt gaat worden zijn deze niet-functionele eisen lastig in te vullen omdat je targets gaat beschrijven onder een bepaalde load van een database. Deze load komt er alleen niet en is ook lastig om te testen. Daarnaast hebben wij er ook voor gekozen omdat security ook een onderdeel is hiervan een aantal niet-functionele eisen toe te voegen waarin beschreven staat hoe we bepaalde dingen gaan uitvoeren terwijl dat niet gebruikelijk is. Bruikbaarheid is ook een lastig onderdeel voor ons omdat wij geen GUI hebben kunnen wij hier weinig over spreken behalve dat er stored procedures zijn die de standaardtaken kunnen uitvoeren zonder hele queries te schrijven. Daarom hebben wij in overeenstemming met onze opdrachtgever voor onszelf een klein aantal niet functionele eisen opgeschreven.

Definition of Done in het Technisch Ontwerp

Wij hebben ervoor gekozen om onze definition of done in het technisch ontwerp te laten staan. De definition of done dient als afspraak tussen de leden binnen de projectgroep voordat een gegeven stuk functionaliteit af is. Hierdoor staan er ook meer technische details in, waar de opdrachtgever niet of minder inspraak over heeft. Om deze redenen zetten we dit onderdeel niet in het plan van aanpak.

CRUD-Matrix in het Technisch Ontwerp

De CRUD-Matrix in dit document stond eerst in ons functionele ontwerp. Na overleg met onze opdrachtgever en begeleider leek het ons handiger om deze in het technisch ontwerp op te nemen sinds hij vooral wordt gebruikt als naslag voor het implementeren van indexes en het nadenken over performance. Omdat de opdrachtgever hier minder inzicht in hoeft te hebben bleek het verwarrender dan nodig als deze in het funtioneel ontwerp stond.

Bronnen

1. Functionele en niet functionele requirements
2. Functioneel Ontwerp