CHECK LIST

* **DEEL 1: Optimalisatie**
  + Afwerken document met optimalisaties
    - De performantie ook ‘globaal’ meten
    - “De DBA heeft geen informatie over de fysieke disks achter de gebruikte volumes. De configuratie kan ook regelmatig gewijzigd worden. Optimaliseer de database hiervoor”
    - Check de tuning advisor na alles te hebben uitgevoerd
    - Zijn er parameters voor de DB engine te wijzigen?
    - Memory optimized tables?
    - Partionering van de tabellen?
    - Compressie toepassen?
* **DEEL 2: Transformaties**
  + Ook nog een tabel voor COUNTRY maken in de DB
  + EVENTUEEL: de 0’en al verwijderen tijdens de XSL transformatie?
  + Zouden we in de XSL ook al meteen de tabellen maken?
  + Er staat in de opdracht dat alles in een JOB steken (zoals de strever) voor teams van 3 is. Dus als we het gewoon 1 maal proper klaar krijgen, ofwel met een JAVA script of met het prog’je, is het prima
    - Laadt het geleverde script in via de job (check hoelang dit duurt)
    - Debug ons script
* **DEEL 3: Datawarehousing**
  + Check of ‘de regels van de kunst’ bewaakt worden
  + de vragen rond de datawarehousing oplossen!
  + In de user-dim wordt nog niet rekening gehouden met de NULL optie (dus geen didicator), de zijn niet in deze versie van de DB maar ze zouden kunnen bestaan in de echte db
  + Check of de juiste velden geinsert worden + test dit in de OP!
  + Check de regen dimensie
    - Check of de rest van de steden op ‘regensituatie onbekend staan’
  + Maak het feit aan!
    - Check alle connecties (met weer, user, etc)
* **DEEL 4: NO SQL**
  + Data creeëren
  + Mongo DB helemaal opzetten
  + Analyse vragen uitwerken

# Probleemdomein

## Inleiding

Het project omvat de analyse en implementatie van een systeem dat grotendeels gelijkaardig is aan dat van GeoCache . Als je het concept nog niet kent kan je er op [www.geocaching.com/guide/](http://www.geocaching.com/guide/) alvast mee vertrouwd geraken. Je zal heel wat concepten herkennen in de opdracht.

Een volledige uitwerking (interviews, use case analyse, domeinmodel, ontwerp, implementatie,…) valt buiten de scope van de opdracht. We concentreren ons op het gegevensbeheer en verschaffen al de nodige artefacten om de opdracht te kunnen uitwerken.

## Probleembeschrijving

Catchem is een bedrijf dat wereldwijd een soort van treasurehunts organiseert. Vrijwilligers verstoppen “Treasures” en plaatsen instructies online om deze “treasures” te vinden. Een treasure heeft op het eindpunt steeds een fysieke container (dat kan ter grootte van een schoendoos maar evengoed veel kleiner zijn). In zo’n container zit een logboek. Als iemand een “treasure” gevonden heeft kan hij een message achterlaten in het logboek en kan hij ook online een logmessage plaatsen.

Catchem heeft veel geregistreerde gebruikers. Van elke gebruiker houden ze volgende zaken bij: **firstname, lastname, street, number, city en email**. De meeste gebruikers zijn geregistreerd om op schattenjacht te kunnen gaan. Als geregistreerde gebruiker kan je ook zelf “Treasures” aanmaken. Je maakt dan zelf een container en plaatst die ergens en registreert dit vervolgens online. Daarna ben je verantwoordelijk admin van die treasure.

Voor elke cache worden **difficulty (0-4),** **terrain (0-4)** en **size (0 NANO ,1 MICRO,2 REGULAR,3 HUGE)**

) bijgehouden. Daarnaast moet de beheerder ook instructies en **één of meerdere stages** toevoegen. Een stage heeft een **geregistreerd coördinaat** (hoogte- en breedtegraad). Om het zoeken op de site sneller te maken wordt aan een treasure de **“City”** opgeslagen van waar deze zich bevindt. Van een “City” wordt ook de **“Country”** opgeslagen.

Er zijn twee verschillende Treasure types .

Een “DirectTargetTreasure” is eenvoudigweg een zoektocht die rechtstreeks naar de “Treasure” leidt. Een “MultiStageTreasure” is een zoektocht die bestaat uit verschillende stages. Gebruikers moeten van één stage naar de andere gaan en vinden bij de laatste stage de uiteindelijke “Treasure”. Een stage kan **"Physical"** of **"Virtual"** zijn. Bij een "Physical" stage is er effectief een container verborgen bij de stage zelf. Bij een "virtual" stage is er enkel een onlinebeschrijving. Meestal is een kenmerk uit de omgeving (bv. een bord van een fietsknooppunt) nodig om te weten te komen wat de coördinaten van de volgende stage zijn. Daarnaast wordt ook de "**visibility" (Visibile = 0, NoCoördinates = 1, Hidden = 2)**van de stage bijgehouden. Als de stage visible is worden voor de schattenjagers zowel de beschrijving als de waypoint getoond. Als de stage "NoCoordinates" als "visibility" heeft wordt de beschrijving getoond maar niet de waypoint. Als de visiblity hidden is wordt ook de beschrijving niet getoond. Bij deze laatste wil dit zeggen dat de pointers waarschijnlijk op de voorafgaande stage beschikbaar zijn. Stages hebben ook een **volgnummer** om de volgtijdelijkheid van de stages vast te leggen.

Als je als user op zoek gaat naar een treasure dan kan naast het fysieke logboek ook een virtuele log registreren. Een log is ofwel een "**found" (=2), "not found (=1)" of "general message (=0)"** log. Bij een log kan je een **bericht** toevoegen en wordt ook de **huidige tijd** opgeslagen. De app kan via het gedrag van de gebruiker (screenviews en GPS bewegingen) ook vastleggen wanneer hij aan de zoektocht is begonnen. **De startTimestamp** wordt ook bijgehouden in de log.

Als het logtype "**found**" is kan je ook informatie over **trackables** meegeven.

Gebruikers kunnen “trackables” kopen en zijn daar vanaf dan eigenaar van. Trackables zijn kleine voorwerpen met een ID op. De gebruiker steekt die in een “cache” zodat een andere gebruiker die kan meenemen en in een andere “cache” kan plaatsen. Op deze manier kan een Trackable de hele wereld rondreizen en kan je hem als eigenaar online volgen. Als je een log "found" registreert kan je daar indien van toepassing bij aangeven welke Trackable je er uit hebt weggenomen en welke je er hebt in geplaatst.

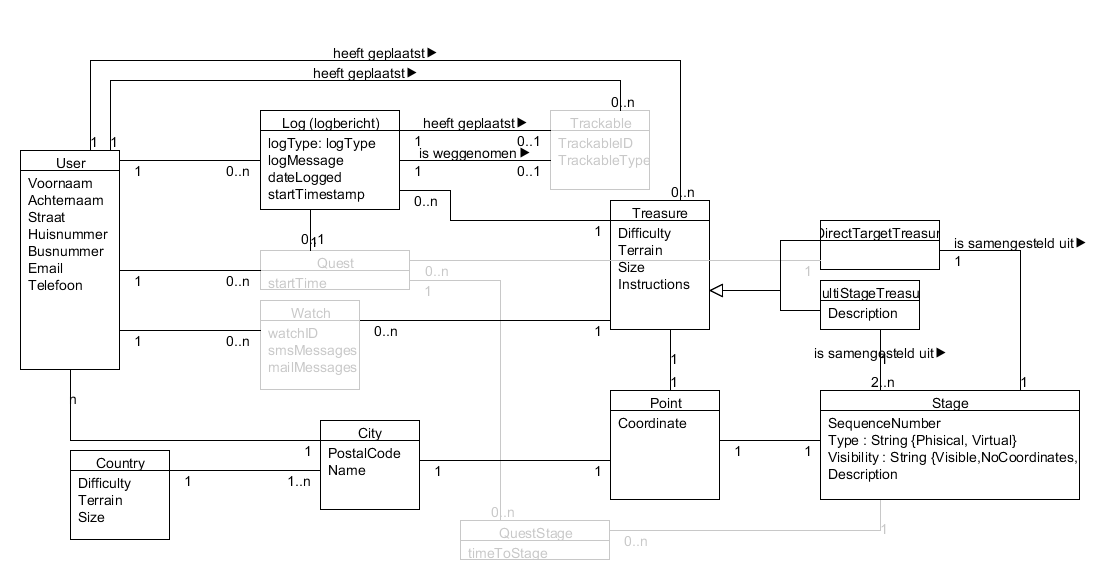
## Later te implementeren functionaliteit

De onderstaande functionaliteit moet nog niet uitgewerkt worden in het huidige project.

(de overeenkomstige klassen in het domeinmodel zijn lichtgrijs)

Je kan als gebruiker ook een TreasureHunt starten bij multiStageTreasures. Bij elke stage wordt ergens een qr code voorzien. Als je met de app een foto trekt van de qr code van de eerste stage start de TreasureHunt. Bij elke Stage wordt de tijd geregistreerd van het moment waarop de foto wordt getrokken. Als je de qr code van de laatste Stage (=Treasure Stage) fotografeert is de TreasureHunt afgelopen. Naargelang je prestaties bouw je punten op.

Als gebruiker kan je ook aangeven of je een bepaalde Treasure wil volgen. Als dat zo is krijg je berichten wanneer er logs worden geplaatst voor die cache. Je kan kiezen via welke kanalen (sms, mail en/of app) je op de hoogte gehouden wil worden.



# Sql Server installeren (DONE)

Voor deze opdracht installeer je Microsoft SQL Server 2017. De installatie instructies kan je op Canvas vinden. Doe deze installatie vooraf thuis en NIET tijdens de les!

# De databank inladen met gegevens

Voor het examenproject vertrekken we van een fysieke databank met daarin testdata. Het grootste deel daarvan wordt volledig ter beschikking gesteld door ons.

### Operationele databank creëren en data inladen

Onder de module optimalisaties vind je het bestand [**Database Restore.pdf**](https://canvas.kdg.be/courses/8099/modules/items/42830) . Hierin staan de instructies om een read-only database restore zoals Catchem\_student.zip opnieuw te herstellen.

# Voor de integriteit te bewaren van de databank tijdens de opzet werd een ‘snap-shot’ gecreëerd van de ingeladen databank om dit snel terug te kunnen herstellen. Dit is te vinden in de SQL file ‘ID transformation’.

# Databank optimalisatie

### Situering

Nadat de database ingeladen is met de nodige gegevens begint het echte werk. De databank heeft namelijk geen enkele optimalisatie meegekregen. De focus in dit deel ligt op het doorvoeren van de nodige databank optimalisaties. Die moeten worden gebaseerd op onderstaande lijst met database vereisten. Voor deze opdracht wordt verwacht dat je een optimalisatie-dossier en de nodige release scripts oplevert. Deze release scripts voeren de door jullie aangebrachte aanpassingen uit op de databank die jullie als startpunt hebben ontvangen. Hoewel we in de leerstof kort ingaan op mogelijke optimalisatietechnieken voor SQL Server is het vooral de bedoeling dat jullie door het bestuderen van de documentatie van SQL Server zelf op zoek gaan naar goede optimalisaties in functie van onderstaande eisen.

**Tip: We hechten veel belang aan de redenering die achter bepaalde keuzes zit. Motiveer daarom voldoende al je keuzes!**

Het optimalisatie-dossier moet volgende zaken bevatten:

* Welke optimalisaties werden onderzocht in functie van onderstaande vereisten.
  + Wat waren de argumenten voor en tegen deze optimalisatie.
  + Welke invloed heeft het de optimalisatie op de performantie?
  + Heb je de optimalisatie toegepast?
  + Instructies over de uitgevoerde acties (in scriptvorm). Detail van welke acties je hiervoor hebt uitgevoerd op de server.

### Database vereisten:

* De database moet geoptimaliseerd worden in functie van de relatieve data hoeveelheden. De verhoudingen van de testdata zullen gelijk blijven bij een toenemend aantal records. Hou er rekening mee dat de dataset in realiteit X maal groter is. Alle gemeenten uit de wereld zijn wel reeds aanwezig. Deze tabel kan dus niet meer groeien.
* Op het systeem zullen er veel lees queries gebeuren maar ook veel inserts en updates.
* De **log tabel is de tabel waarin de meeste nieuwe rijen worden aangemaakt**.
* Het systeem zal (zowel via de site als via de app) over volgende schermen beschikken:
  + Gebruikers kunnen zoeken op **treasure hunts** in de buurt waarbij je kan filteren op **moeilijkheid, reliëf en aantal stages**
  + Je kan detail opvragen van **een treasure** waarbij je volgende informatie ziet:
    - de laatste **X geplaatste logberichten**
    - **Beschrijving**
    - Gegevens van de gebruiker waaronder **het aantal beheerde treasures**
    - Detail van de stages getoond naargelang de ingestelde **zichtbaarheid van de stages**.
  + De **helpdesk kan ook lijsten opvragen van gebruikers**
  + De helpdesk heeft een scherm ter beschikking om logs te **modereren en daarvoor te zoeken op de inhoud van de log beschrijvingen**.
  + Uiteraard kunnen **gebruikers treasures (met de daarbij horende stages) aanmaken en logs aanmake**n. Bij het registreren wordt dan weer een gebruiker aangemaakt.
* De DBA heeft geen informatie over de fysieke disks achter de gebruikte volumes. De configuratie kan ook regelmatig gewijzigd worden. Optimaliseer de database hiervoor.

### Hoe ga je te werk?

Lees aandachtig de database vereisten. Probeer je database te optimaliseren **zodat je de gebruikers van het systeem de meest optimale gebruikservaring geeft**.

Zorg ervoor dat je je optimalisaties kan testen. Maak queries die vaak gebruikt zullen worden en gebruik deze om de optimalisaties te testen. Maak veelvuldig gebruik van de execution plans en vergeet ook zeker de Tuning Advisor niet te gebruiken.

Je kan jezelf heel wat vragen stellen. Voorbeelden zijn: Zijn er parameters voor de database engine die ik kan wijzigen? Welk type indexen gebruik ik waar? Welke datatypen zijn optimaal? Kan ik best memory optimized tables inzetten, en waar? Is het nuttig om op bepaalde tabellen partitionering toe te passen? Is het nuttig om op bepaalde tabellen compressie toe te passen? Welke optimalisaties die de docent besproken heeft zouden nuttig kunnen zijn?

# Evaluatiecriteria

* De professionaliteit van het optimalisatie dossier
* De gekozen optimalisatietechnieken in functie van de vereisten
* De verdediging van de gekozen optimalisatietechnieken
* De wijze waarop de invloed van bepaalde optimalisaties werd getest en al dan niet doorgevoerd

**~~Niet voor teams met 2 personen:~~**

## ~~Scheduled backup~~

~~Zorg ervoor dat je dagelijks een backup maakt van de databank. De backup files dienen zo compact mogelijk te zijn.~~

~~Je moet kunnen aantonen dat je de databank gewoon kan restoren na een probleem met de database.~~

## ~~Beveiliging~~

~~Zorg ervoor dat je voor volgende profielen de juiste rollen aanmaakt in de databank. Test datgene dat je hebt uitgewerkt. Het is belangrijk dat niemand meer kan doen dan datgene dat hij hoort te doen.~~

~~- Rol voor datawarehouse gebruiker~~

~~- Rol voor ingeschreven gebruikers (diegene die op schattenjacht gaan of zelf schatten beheren)~~

~~- Rol voor de software die het datawarehouse voorziet van data. Die data is gebaseerd op de gegevens uit het operationeel systeem.~~

~~- Rol voor een administrator die gebruikergegevens volledig kan beheren.~~

**Deel 2: ETL Transformations**

We nemen de XML file [**CountryCityList.zip**](https://canvas.kdg.be/courses/8099/modules/items/43101) . Hier staan alle landen en hun steden in.   
We zullen in Week 2 zien hoe je de transformatie van XML naar SQL server insert statements kan aanpakken,

**Tip**: De xml file is wat te groot om te bekijken in een normale editor. Zelfs Notepad++ heeft er problemen mee. Ik heb een lightweigt programma gevonden die de file vrij snel open krijgt   
( <http://www.firstobject.com/dn_editor.htm> )

**Enkele opmerkingen:**

* Je moet eerst de XML unzippen 47Mb → 600Mb.
* Het gegevens city\_id in de xml verwijst naar 2 tabellen:
  + Treasure.city\_city\_id
  + user\_table.city\_city\_id
* de city\_id is opgeslagen als **BASE64**, dit type komt niet overeen met het type in de databank. Zoek hier een weg rond.
* Als de grootte van de file in het begin een probleem vormt, hebben we je 2 hulpmiddeltjes gegeven:
  + [**country\_list\_sample.xml**](https://canvas.kdg.be/courses/8099/modules/items/43937)

Dit is een sample van de voorbeeldfile, er zit een deel van 2 landen met hun steden in, je kan hier al mee starten om je transformatie te testen, zonder dat je direct met veel geheugenproblemen zit.

* + [**country\_list\_sample - pretty.xml**](https://canvas.kdg.be/courses/8099/modules/items/43936)  
    Dit is een voorbeeldfile waar we de XML in een meer leesbaar formaat hebben gezet, dit bevat een beperkte set van 2 landen. Omdat de hier meer whitespace tussen zit, zou ik deze file niet als input gebruiken van je script.

In de databank ontbreken **de landen** **en steden** op dit moment. Via een externe dienst worden **deze periodiek** aangeleverd in XML formaat. Download de meest recente versie: **CountryCityList.zip** .

## De transformatie uitwerken

Het bestand dient omgezet te worden naar twee tabellen in de databank. Maak hiervoor een XSL bestand dat SQL statements genereert.

**Studenten die individueel werken: Je moet enkel in staat zijn insertscripts te maken met het sample bestand. Er wordt op Canvas een extra db backup bestand voorzien waarin de Country en City tabel reeds inzitten zodat je ze niet moet uitwerken.**

**Teams van 3: Zorg ervoor dat dit alles in een Pentaho Job wordt uitgevoerd waarbij je eerst het SQL script aanmaakt en daarna dat script ook effectief uitvoert.**

## Opmerkingen

* De xml file is wat te groot om te bekijken in een normale editor. Zelfs Notepad++ heeft er problemen mee. Met volgende lightweight programma krijg je het toch vrij een voudig open: h[ttp://www.firstobject.com/dn\_editor.htm](http://www.firstobject.com/dn_editor.htm)
* Je moet eerst de XML unzippen 47Mb → 600Mb.
* Het gegevens city\_id in de xml verwijst naar 2 tabellen:
  + Treasure.city\_city\_id
  + user\_table.city\_city\_id
* de city\_id is opgeslagen als BASE64, dit type komt niet overeen met het type in de databank. Zoek hier een weg rond.
* Werk eerst met volgende hulpbestanden. Die zijn kleiner en daardoor voor het testen van je code veel eenvoudiger te gebruiken.
  + **country\_list\_sample.xml**

Dit is een sample van de voorbeeldfile, er zit een deel van 2 landen met hun steden in, je kan hier al mee starten om je transformatie te testen, zonder dat je direct met veel geheugenproblemen zit.

* + **country\_list\_sample-pretty.xml**  
    Dit is een voorbeeldfile waar we de XML in een meer leesbaar formaat hebben gezet, dit bevat een beperkte set van 2 landen. Omdat de hier meer whitespace tussen zit, zou ik deze file niet als input gebruiken van je script.

# Evaluatiecriteria

* Je kan het XML bestand correct transformeren met behulp van XSL
* Je kan omgaan met onvoorziene problemen bij de conversie (vb. groot bestandsformaat)

Momenteel gaan we voor een oplossing waar we eerste de XLM splitsen (op land) met het programma firstobject (h[ttp://www.firstobject.com/dn\_editor.htm](http://www.firstobject.com/dn_editor.htm)). Dit halen we dan in batch (via pentaho) door een XSL transformatie die het wegschrijft naar een .SQL file. Tenslotte wordt dit dan in de database geladen door een SQL stap in Pentaho.

**Deel 3: DATAWAREHOUSING**

# Situering

Catchem wil analyses trekken rond de logs die geregistreerd werden voor de Treasures**.** Er werd besloten een DWH op te zetten en als pilootproject een ster rond de **subject area TreasureFound** te bouwen. In onderstaande diagram zie je wat het analistenteam geschetst heeft. Bouw op basis van dit beperkte model de ster en vul deze met data uit de correcte brontabellen met behulp van de ETL-tool. De data in de ster beperkt zich tot data van ‘Found’ logs. **Hou rekening met de regels van de kunst en zorg ervoor dat op basis van de inhoud van de tabellen propere en leesbare rapporten gemaakt kunnen worden.**

Lees eerst de opgave in zijn geheel door zodat je een zicht hebt op wat er allemaal van je verwacht wordt.

Nadat je het datawarehouse correct hebt gevuld moet je analyse query’s opstellen voor een aantal analysevragen. Naast de query’s voor de hieronder reeds opgesomde analysevragen voorzie je zelf 4 bijkomende analysevragen met bijhorende query.

**Studenten die individueel werken: Analysevragen in verband met het weer moeten niet uitgewerkt worden.**

* Welke rol hebben datumparameters (dagen, weken, maanden, seizoen) op het aantal caches?
* Wat is de invloed van het type user op de duur van de treasurehunt? Doet een beginner er langer over?
* Vinden users de cache gemiddeld sneller in de regen?
* Zoeken beginnende users gemiddeld naar grotere caches?
* Worden er gemiddeld minder caches gezocht op moeilijker terrein als het regent?
* Worden er in weekends meer moeilijkere caches gedaan?

# 

# Detail van het feit en de dimensies

## De datum dimensie

GeoCache wil statistieken kunnen trekken op basis van seizoenen. Voorzie in de datumdimensie (hiervoor heb je al een basis) één veld dat aangeeft welk seizoen het is. We gaan er voor de eenvoud vanuit dat er enkel op het noordelijk halfrond aan GeoCaching gedaan wordt. Je kan zelf kiezen hoe je dit aanpakt (Spoon steps gebruiken om de seizoenen te berekenen, extra tabel aanmaken…)

Wij hebben dit gedaan door in pentaho het seizoen te berekenen (via een function stap) dmv de maanden te gebruiken (metrologische seizoenen die beginnen aan het eind van de maand).

## De user dimensie

Geocache wil kunnen rapporteren op een aantal belangrijke zaken die gekoppeld zijn aan de gebruikers. In de dimensie moet naast de Voornaam, Achternaam, Woonplaats en Land ook bijgehouden worden wat de ‘ExperienceLevel’ is. Volgende levels komen voor:

* Newbie: Nog geen ‘Found’ logs geplaatst
* Amature: Minder dan drie ‘Found’ logs geplaatst
* Regular: 3-10 ‘Found’ logs geplaatst
* Expert: Meer dan 10 found logs geplaatst

Daarnaast wordt ook bijgehouden of hij al dan niet een ‘Dedicator’ is. Dit is het geval als hij zelf beheerder is van minimum één Cache

Voor woonplaats, land, experiencelevel en dedicator wil Geocache bij analyses steeds de situatie op het moment van de logdatum kunnen raadplegen. Als een gebruiker op dag d experiencelevel:Regular had en op dag d+1 experiencelevel:Expert dan moet een CacheFoundEvent van d+1 gekoppeld worden aan de dimensieversie van de gebruiker met experiencelevel:Expert. Voor een CacheFoundEvent van dag d moet dat Regular zijn.

Wij hebben dit uitgevoerd door in Pentaho 2 table inputs te combineren in een merge step en een dmv een formula step de experience level te berekenen en aan te geven dat de gebruiker een dedicator is of niet.

Verder werd een SDC TYPE 2 transformatie opgezet. Dit zorgt ervoor dat veranderingen in de DB zullen geregistreerd worden als ‘versie 2’ bij bepaalde velden.

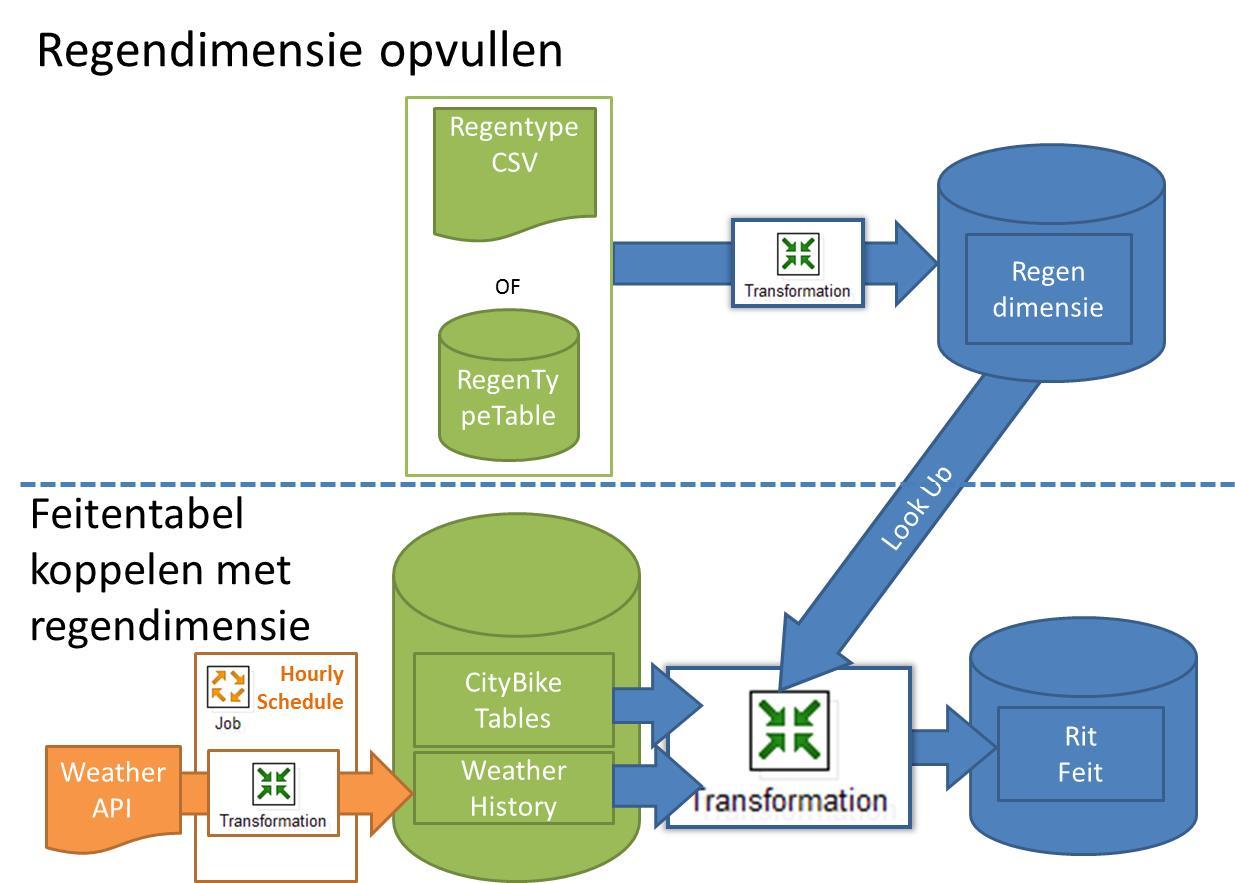
## De regendimensie

**Studenten die individueel werken: De weerdimensie en de historiektabel die hieraan gekoppeld is moet niet uitgewerkt worden.**

Ook het weer kan een belangrijke parameter zijn voor de analyse van gevonden treasures. Voorzie een regendimensie die de verschillende regentypes bevat. Je moet deze koppelen aan het feit op basis van de regensituatie op het moment van de logdatum. Maak om dit te verwezenlijken gebruik van een weer-api.

Let op: Het aantal rijen in de regendimensie is beperkt tot weertypes met regen in de weer-api (zie <http://openweathermap.org/weather-conditions>) + 1 rij voor alle weertypes zonder regen + 1 rij voor "Regensituatie onbekend". De bron van de regendimensie kan je zelf maken. Dit kan bijvoorbeeld via CSV of een tabel.

Informatie over hoe je de regensituatie op moet halen kan je vinden bij "Het feit TreasureFound".



## De dimensie TreasureType

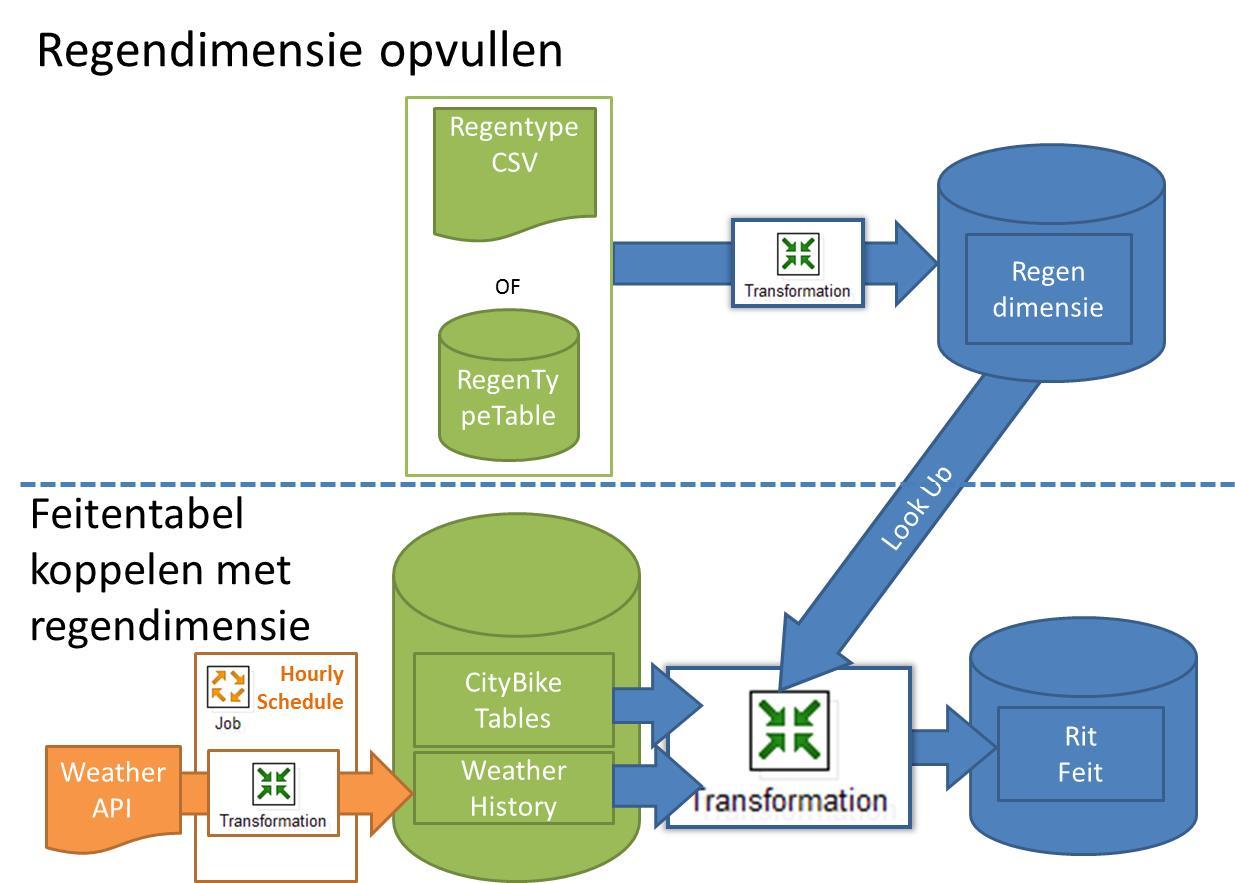
In de dimensie treasureType worden gegevens bijgehouden over de cache die gevonden is. Van een cache wordt volgende informatie in de dimensie opgeslagen: **Difficulty, Terrain, Size** (aantal stages binnen een treasure)

## Het feit TreasureFound

Centraal in de ster staat de feitentabel **TreasureFound**. Voor elke gevonden treasure wordt hier een record in opgeslagen. Je kan een treasure als gevonden beschouwen als er een log **van het type ‘Found’ (= 2)** voor werd geregistreerd. Zorg voor de koppelingen met de dimensies die hierboven beschreven staan, een **standaard meetwaarde (met telkens een 1 als waarde**), een **meetwaarde voor de duur van de quest** en zorg ook voor een veld met de **creatiedatum van het record**. Zorg voor de juiste koppeling met **de type 2 dimensie ‘User’ en de andere dimensies**

**Studenten die individueel werken: De weerdimensie en de historiektabel die hieraan gekoppeld is moet niet uitgewerkt worden.**

Let op: In het bronsysteem is geen weersinformatie aanwezig. Je hebt hiervoor dus een bijkomende bron nodig.   
Je zou historische weersinformatie kunnen opvragen via een weer-api, maar dat kost geld.   
Bouw daarom op basis van de huidige weersituatie zelf historiek op. Maak hiervoor een transformatie in DI die als hij wordt uitgevoerd per stad een rij toevoegt aan een historiektabel met daarin het uur de dag en de stad van de waarneming en de waarneming zelf. Je beperkt je tot 10 steden uit de database (de api's hebben nogal strikte limieten). De weergegevens kan je bijvoorbeeld van <http://openweathermap.org/current> halen.



Als je voor een bepaalde stad voor dat uur geen regeninformatie kan vinden in de historiektabel dan zet je voor die record het regentype op "Regensituatie onbekend".

Maak ook een job die de feitentabel leegmaakt en die vervolgens terug helemaal opvult.

*Tip: Als het uitvoeren van de job te lang duurt voeg dan een logische filtering toe in je query bij het inlezen uit het bronsysteem. Je kan die nadat alle transformaties getest zijn terug weghalen om een volledige run te doen. Lang wachten tijdens testruns is immers verloren tijd.*

## ~~Optimalisatiedossier (Niet voor teams met 2 leden of individuele studenten)~~

~~Vul het optimalisatiedossier uit opdracht deel 2 aan met een hoofdstuk rond datawareoptimalisatie. Ga hiervoor even grondig te werk maar zoek naar optimalisaties die voor de structuur van de datawarehouse databank extra goed geschikt zijn.~~

## ~~Extra dimensie (Niet voor teams met 2 leden)~~

~~Voorzie een extra dimensie in het datawarehouse. Zoek zelf naar een nuttige dimensie. Kies het gepaste SCD type en werk de dimensie mee uit in Pentaho DI. Zorg ook voor 3 bijkomende analysevragen die gebruik maken van deze dimensie.~~

# Evaluatiecriteria

* De werking van de ETL-transformaties en jobs in DI.
* De maten waarin aan de analysebehoeften uit de opdracht is voldaan.
* De volledigheid van de query’s die een antwoord bieden op de analysevragen.
* De maten waarin de richtlijnen uit de theorie gevolgd werden bij de uitwerking van de ster.

# DEEL 4: NoSQL

Evoluties zoals ‘The internet of things’ en ‘Big data’ openen nieuwe mogelijkheden voor het vergaren en analyseren van gedetailleerde informatie uit fysieke objecten.

Catchem wil extra inkomsten genereren. Om een extra dimensie te geven aan de zoektocht naar treasures introduceren ze een smartwatch met de naam CatchemAll. Deze watch heeft een uitgebreid scala aan sensoren waardoor er leuke extra informatie ontstaat rond de zoektocht naar treasures. Elke maand organiseert Catchem hiermee een wedstrijd waar ze telkens de beste treasurehunters uitdagen, de beste wint dan een prijs.

De watch beschikt over volgende sensoren:

* Hartslagmeter (bpm)
* Altimeter (decimeter)(hoogtemeter)
* Temperatuurmeter (Celsius)
* DecibelMeter(dB)

Enkele voorbeelden van de wedstrijden.

* Zoek treasure na treasure totdat je in totaal min. 10 km hebt afgelegd. De winnaar is degene die gemiddeld de laagste hartslag had tijdens die quests.
* De treasurehunter die 10 treasures kan vinden met de grootste totale klimhoogte
* Zoek 8 treasures en doe dit zo stil mogelijk.

Een gebruiker kan via zijn watch een quest naar een single stage treasure starten als hij binnen een kilometer van een Treasure stage is. Vanaf dan worden **real-time meetgegevens in JSON formaat** uitgestuurd naar een centrale service, waar ze in **een MongoDB collection worden opgeslagen voor verdere analyse**.

Elke 10 seconden worden er meetgegevens doorgestuurd samen met ***de ID van de gebruiker, de ID van de quest, de ID van de stage, datum/tijd,de GPS coördinaten* (long/lat) en afstand van de quest.**

(Voor de vaste gegevens start je van een gps coördinaat **op maximum 1 kilometer van je stage-locatie, elke 10 seconden verplaats je random de huidige locatie een beetje dichter, voor de afstand begin je op 0 en zet je de afgelegde waarde tussen coördinaat X-1 en coordinaat X, het is een ‘nice to have’ om deze echt te berekenen, indien je dit niet direct vindt, kan je je getal random laten verhogen).** Omdat niet alle sensoren elke seconde een succesvolle meting kunnen uitlezen worden niet steeds de resultaten van alle sensoren doorgestuurd.

We maken abstractie van alle aspecten die komen kijken bij het effectief versturen en ontvangen van data sensor data. Het startpunt is een lege MongoDB collectie.

Zorg voor de nodige data in je collectie volgens onderstaande specificaties. Daarna voer je nog de opgelegde analyses op die data uit.

## Opvullen van de collectie

Voor het opvullen van de collectie gebruik je JSON berichten afkomstig van je eigen test sensor. Hou rekening met volgende aannames:

1. Zorg voor beperkt en hanteerbaar aantal testberichten waarop je het opvullen baseert. Mik op ongeveer **10 000 berichten voor min. 200 quests, verspreid over 2 direct target treasures.** Als je de testdata random genereert via een zelfgeschreven applicatie is het uiterst verstandig meer testdata aan te maken.
2. De **quests vinden allemaal op dezelfde dag** plaats.
3. Het aantal sensoren dat per meting uitgelezen wordt (1,2,3 of 4) is gelijkmatig verdeeld. Welke combinatie van sensoren dat zijn is volledig willekeurig.
4. Alle parameterwaarden zijn anders geschaald[[1]](#footnote-1) en er zijn ook beperkingen op de variatie van de uitgelezen waarden:
   * Hartslag
     + Schaal ligt **tussen 60 en 200 bpm**
     + Kan vrij variëren binnen een quest
   * Altimeter
     + Tussen -**100 en 100.000 dm**, men kan niet meer dan 50 meter per 10 seconden in hoogte bewegen.
   * Temperatuur
     + Varieert tussen -**50,0 en +50,0 graden** celsius nauwkeurig op 1 tiende graad.
   * DecibelMeter
     + Varieert tussen **0 en 150 Db.**
5. De GPS-coördinaten worden opgeslagen tot op 6 cijfers na de komma wat overeenkomt met een nauwkeurigheid van ongeveer 1 meter. Elke meting heeft een geldig GPS coördinaat.  
   Let er op dat je binnen een quest je niet meer dan 1000 meter van de treasure Stage bevindt (je moet dus wel eerst de locaties van de stages vastleggen). We negeren het feit dat je in principe in één seconde geen 100 meter kan afleggen. De coördinaten mogen dus random zijn.
6. Afhankelijk van het aantal teamleden van het project:
   * Voor teams met 2 personen: De Treasures, quests en gebruikers die je gebruikt moeten niet overeenkomen met die in de operationele databank.
   * ~~Voor teams met 3 personen: Werk met bestaande stage-id’s en gebruiker-id’s die uit je eigen database komen. Zorg er ook voor dat je data logisch in de tijd wijzigt:~~
     + ~~Sensordata varieert incrementeel tussen twee metingen: Zorg er dus voor dat de temperatuur niet in één seconde met 3 graden stijgt.~~
     + ~~GPS coördinaten eindigen op het einde van de quest op de plaats van de laatste treasure stage.~~
     + ~~De hartslagwaarden stijgen naarmate je dichter bij de cache komt.~~

Het opvullen mag gerealiseerd worden vanuit een generator naar keuze (vb. excel, java of dotnet prog, pentaho di, mockaroo...)

De gegevens van de Treasure zelf hoef je niet te bewaren in je document database, maar zorg dat de quests die aan dezelfde treasure koppelt dezelfde treasure-id hebben.

*Opgelet: als je mongoimport gebruikt: deze kan timestamps enkel herkennen in het standaard ISO 8601 formaat via een geneste $date: “<string in ISO format>” constructie (zie handleiding).*

## Management

Zorg ervoor dat je sharding toepast met minstens twee shards (mongod instanties). Kies een goede shardkey.

Je mag dit lokaal uitwerken door twee mongos op je machine te laten draaien.

Teams met 3 personen moeten ook replica sets definiëren met minstens 2 replica’s per set. Bekijk in de tutorial voor meer uitleg over hoe dit op te zetten. Het is mogelijk dat je dit niet op één machine kan uitvoeren, extra shards moet voorzien, extra mongos configureren, ...

## Analyse van de data uit de collectie

Geef een antwoord op volgende analyse vragen. Doe dit met behulp vanscripting[[2]](#footnote-2) (query’s/aggregation pipeline)

**Voor individuele studenten: De 5de analysevraag moet niet uitgewerkt worden.**

1. Geef een overzicht van alle metingen gerangschikt per treasure en daarbinnen per quest.
2. Geef per quest de gemiddelde hartslag weer.
3. Geef een overzicht van alle metingen voor quests bij een gemiddelde hoogte per quest hoger dan 300 meter uitgevoerd zijn.
4. Neem één centraal punt en geef dan een overzicht van alle GPS-coördinaten die gemeten zijn in een straal van 100 km (dit mag met flat of spherical 2D gedaan worden)

Voor teams van 2 personen:

1. Voorzie Heat map[[3]](#footnote-3) support: Op het internet willen we rond een stage cirkels (10 meter, 20 meter, 30 meter , 40 meter, 50 meter rond de cache) kunnen tonen. Een cirkel krijgt een kleur naarmate de hoogte van de meting. Maak hiervoor een collectie aan in MongoDB waar je per stage en per straal de gemiddelde waarden van de Decibelmeter (Db) en Hartslag (bpm) opslaat. De heatmap mag per stage afzonderlijk aangemaakt worden.

Voor teams van 3 personen:

1. Schrijf queries voor elk van de voorbeelden van de wedstrijden
   1. Zoek treasure na treasure totdat je in totaal min. 10 km hebt afgelegd. De winnaar is degene die gemiddeld de laagste hartslag had tijdens die quests.
   2. De treasurehunter die 10 treasures kan vinden met de grootste totale klimhoogte
   3. Zoek 8 treasures en doe dit zo stil mogelijk.

(Voor bepaalde vragen mag je je dataset uitbreiden zodat de data aan de vragen beantwoord, misschien voor extra punten)

1. Bedenk zelf nog een mogelijke wedstrijd (variatie op de voorbeelden) en schrijf een query die de winnaar aanduidt gebaseerd op je gegenereerde data.

## Evaluatiecriteria

* Correcte opvulling en analyse van sensor berichten
* Inzicht in de concepten aangebracht in de NoSQL slides
* Vlot kunnen werken met MongoDB (management, scripting,..)

1. In mockaroo kan je gebruikmaken van scenario’s om dit te verwezenlijken. Je moet daar wel voor ingelogd zijn. [↑](#footnote-ref-1)
2. Je kan gerust Robo3T en de daarin ingebouwde shell gebruiken om alles uit te testen. De finale oplevering bestaat uit script file(s). [↑](#footnote-ref-2)
3. Een ‘heatmap’ is een weergave op een kaart van een bepaalde parameter waarbij kleur (rood=hoog, blauw=laag) wordt gebruikt om de waarden weer te geven [↑](#footnote-ref-3)