Algemeen:

De bedoeling van dit blok is dat je aan het einde een relationele database *efficiënt* kunt gebruiken, eventueel in combinatie met Python.

Het is de bedoeling dat je individueel werkt en inlevert. Dat betekent overigens niet dat je niet mag overleggen of elkaar niet mag helpen. Het betekent wèl dat als ik antwoorden tegenkom die (nagenoeg) kopieën van elkaar zijn, ik beide antwoorden fout reken.

Beoordeling:
Voor de deelopdrachten krijg je punten die bij elkaar opgeteld samen maximaal negen punten van je eindcijfer vormen. Bij het aantal punten dat je voor je opdrachten verzameld hebt tel je één op voor je uiteindelijke punt. Bij een vijf-en-een-half of hoger heb je een voldoende.

Inleveren opdrachten:
De opdrachten lever je in door je werkstuk in de daarvoor aangemaakte inlevermap op Blackboard te uploaden. Je werkstuk heeft altijd de volgende naam (waarbij de strings tussen <> verplicht zijn om door jou in te vullen met jouw eigen gegevens):
`<voornaam>_<achternaam>_deelopdracht<nummer>.<extensie>`

Als ik zelf bijvoorbeeld deelopdracht 01 in zou leveren dan zou ik dat moeten doen door een .zip aan te maken met de naam 'Ronald_Wedema_deelopdracht01.zip', om deze daarna in de Blackboard inlevermap te plaatsen.

Bestaat het werkstuk dat je voor een deelopdracht in moet leveren uit meer dan één bestand, dan verpak je de bestanden in een .zip die je benoemt en inlevert zoals hierboven beschreven.

Te laat of foutief ingeleverde opdrachten worden niet geaccepteerd. In goed overleg valt er natuurlijk een enkele keer een deadline op te schuiven.

De deelopdrachten zijn bedoeld om je stap voor stap kennis te laten maken met het gebruik in de praktijk van relationele databases. Mis je een deadline en dus de punten, dan is het nog steeds verstandig dat je de opdracht maakt omdat je je daarmee voorbereidt op de volgende deelopdracht. De week na de deadline wordt er klassikaal feedback op een deelopdracht gegeven. Als iets niet duidelijk is, vraag het dan, anders heb je verderop in het traject waarschijnlijk weer een probleem.

Tenzij ik expliciet iets anders vraag is alle code voorzien van (pydoc) commentaar en object georiënteerd en opgebouwd volgens de lagenstructuur waarmee je in de eerste les kennis maakt.

Tijdens de lessen is er ruim gelegenheid om aan de opdrachten te werken en mij om ondersteuning of uitleg of persoonlijke feedback op gemaakte opdrachten te vragen. Maak daar gebruik van!
Voor dit vak krijg je 3 EC uitgekeerd. Dat wil zeggen dat de gemiddelde student ongeveer 3 * 27 uur aan dit vak moet besteden. Van die 81 uur zijn er 23 ingepland als contacttijd. De rest moet je *zelf* invullen.

Mocht je onverhoopt aan het einde toch minder dan een 5,5 gehaald hebben, dan krijg je van mij een reparatieopdracht.

deelopdracht 01:
Maximaal 1.0 punt.
Deadline 25-11-2019 23:59:59
Inleverformat: een .zip met de sources en een README.txt met daarin de exacte syntax waarmee ik het script in de terminal kan starten.

Gebruik voor deze opdracht de database die je met *studenten.sql* aanmaakt. Dit bestand kun je vinden onder de Databases2 course op Blackboard.

Normering:
scripts die niet werken, niet OO zijn of niet gedocumenteerd: 0 punten.
scripts die slecht gedocumenteerd zijn maar wel aan de overige eisen voldoen: maximaal de helft van het maximaal aantal punten, naar beneden afgerond op tienden.
voor inefficiënte of slecht gestructureerde code kun je, ook als het werkt, maximaal de helft van het maximaal aantal punten krijgen, naar beneden afgerond op tienden.
Als ik het bestand niet kan lezen krijg je 0 punten.

Opdracht 1 (0.4 pnt)
Maak een python module aan waarin je een klasse 'Tentamen' definiëert die:

1. in zijn constructor waardes voor de properties 'student', 'vak', 'datum' en 'cijfer' meekrijgt
2. die, als je een object van deze klasse wil printen, het volgende uitprint (let op de uitlijning, tussen < en > staat natuurlijk de waarde van de property):

```
student:  <student>
vak:      <vak>
datum:    <datum>
cijfer:   <cijfer>
```

Opdracht 2 (0.4)
Maak een python module die:

1. m.b.v. *mysql.connector* de toegang tot je database regelt
2. een method heeft die de namen van alle studenten in een list teruggeeft
3. een method heeft om aan de hand van de naam van een student(v/m) zijn/haar resultaten op te halen en deze opslaat in tentamen objecten, geef deze tentamen objecten terug in een lijst of een dictionary.

Opdracht 3 (0.2 pnt)
Maak een python script dat standaard de namen van alle studenten laat zien. Door de naam van een student als parameter mee te geven moeten alle tentamens die door die student gemaakt zijn getoond worden. Dit script gebruikt de module uit opdracht 2.

deelopdracht 02:
Maximaal 0.5 punt.
Deadline 04-12-2019 23:59:59
Inleverformat: een .zip met de source van je database module en een README.txt met daarin de exacte syntax die nodig is om je script te draaien.

Normering: zie Deelopdracht 1.

Opdracht 1 (max. 0.4 pnt)
Breid de python module uit zodat sql-injectie onmogelijk wordt.

Opdracht 2 (max. 0.1 pnt)
Laat de module de inloggegevens voor de database uit een configuratiebestand halen dat “my.cnf” heet en in dezelfde directory als het script staat.
De inhoud van dit bestand is gelijk aan de .my.cnf die standaard gebruikt kan worden door de client applicaties van mySQL.

deelopdracht 03:
Maximaal 2.0 punt.
Deadline 11-12-2019 23:59:59
Inleverformat: een .zip met een pdf en een sql-script.

Opdracht 1 (max. 1.0 pnt)
Maak een ontwerp (een entiteiten diagram), zoals je dat in databases1 geleerd hebt, van een relationele database die gebruikt kan worden om de gegevens uit de volgende informatiebehoefte in op te slaan. Geef ook een tekstuele toelichting *wat* je *waarom* opslaat en *wat* je *waarom* weg laat.

Een (eukaryoot) organisme heeft chromosomen waar op zich de genen van dat organisme bevinden. Een gen heeft altijd een sequentie en minimaal één identifier waaronder hij te vinden is in de internationale databases en is onder te verdelen in korte oligonucleotiden. Eigenschappen van oligonucleotiden zijn, bijvoorbeeld, het gc-percentage en de smelttemperatuur.
Op een microarray bevinden zich duizenden probes, elk op een vaste positie. Elke probe is in feite een oligonucleotide met een lengte van 25 basen. Een oligonucleotide moet verder nog aan de volgende voorwaarden voldoen om als probe gebruikt te kunnen worden:

- de sequentie is uniek binnen het coderende genoom
- de sequentie bevat geen repeats van meer dan vier nucleotiden
- dinucleotide repeats mogen maximaal twee nucleotide-paren groot zijn
- de oligo mag geen hairpinstructuren kunnen vorasci convertmen

Binnen een microarray mogen probes natuurlijk niet overlappen en moeten de smelttemperaturen van alle probes binnen bepaalde grenzen van de incubatietemperatuur van de array liggen. De kwaliteit van de array hangt er van af hoe ver de incubatietemperaturen van de gebruikte probes maximaal uit elkaar liggen en welk percentage van de genen tenminste één probe op de array heeft liggen en welk percentage van de genen meer dan één probe op de array hebben. Een oligo kan natuurlijk op meerdere arrays als probe gebruikt worden.

Normering:

- voor elke ontbrekende tabel 0.3 punten afttrek
- voor elk ontbrekend of misplaatst veld of datatype 0.1 punten aftrek
- voor elke ontbrekende of foute foreign key 0.2 punten aftrek
- voor elke ontbrekende primary key 0.3 punten aftrek
- overige fouten naar bevinding

Opdracht 2 (max. 1.0 pnt)
Maak een sql script dat je ontwerp implementeert.

- Zorg voor goed commentaar.
- Zorg er ook voor dat de tabellen volledig verwijderd worden als ze al bestaan als het script uitgevoerd wordt.
- Zorg voor goede defaultwaarden.
- zorg voor duidelijk commentaar

Normering: zie deelopdracht01. (N.B. het object georiënteerd zijn is hier natuurlijk niet van toepassing)

deelopdracht 04:
Maximaal 2.0 punt.
Deadline 19-12-2019 23:59:59
Inleverformat: een .zip met de source voor de database module en een compleet sql script en een databestand.

Normering: zie deelopdracht01 en -03

Opdracht 1 (max. 0.5 pnt)
Breid het laatste sql script uit met een bulk import. Heb je er al meer in staan commentaar die dan uit.
Voeg een voorbeelddatabestand bij zodat ik het script ook kan testen. Er moeten minimaal één chromosoom, twee genen en vijftig oligo's toegevoegd worden. Dit mogen dummies zijn.

Opdracht 2 (max. 0.5 pnt)
- Breid het laatste sql script uit met een stored routine, sp_get_genes, die de externe id en de sequentie van alle genen op haalt.
- Breid het laatste sql script uit met een stored routine, sp_get_tm_vs_probes, die het aantal verschillende smelttemperaturen gedeeld door het aantal oligo's/probes ophaalt.
- Breid je sql script uit met een stored routine, sp_mark_duplicate_oligos, die oligo's/probes waarvan de sequentie niet uniek is binnen het coderende genoom als zodanig kenmerkt. De benodigde informatie krijgt deze stored procedure mee als parameters. Ook deze stored routine moet te benaderen zijn door een method in je python module.

Opdracht 3 (max. 1.0 pnt)
Maak een nieuwe python module die m.b.v. mysql.connector de toegang tot een database regelt. Verwerk hierin alles wat je dit blok al geleerd hebt. De bovenstaande stored routines zijn methods van de klasse in deze module.

deelopdracht 05:
Maximaal 2.0 punt.
Deadline 07-01-2020 23:59:59
Inleverformat: een .zip met een compleet sql script, een databestand en een pdf.

Opdracht 1 (max. 1.0 pnt)
Maak een pdf waarin je duidelijk in correct nederlands uitlegt waarom welke indices nodig zijn voor je database, welke indices misschien nuttig zijn (en waarom) en welke indices ongewenst zijn en waarom.

Normering: Punten afhankelijk van correctheid van de redeneringen, ter mijner beoordeling. Voor incorrect Nederlands worden punten afgetrokken.

Opdracht 2 (max. 1.0 pnt):
Breid het laatste sql script uit met de gewenste indices.
Voeg een voorbeelddatabestand bij zodat ik het script ook kan testen. Er moeten minimaal één chromosoom, twee genen en vijftig oligo's toegevoegd worden. Dit mogen dummies zijn.

Normering: zie deelopdracht01 en -03

deelopdracht 06:
Maximaal 2.5 punt.
Deadline 15-01-2020 23:59:59
Inleverformat: een .zip met de source voor de database module en een compleet sql script en een databestand.
Het bestand moet leesbaar zijn door de groep, maar niet voor 'others'.

Normering: zie deelopdracht01 en -03.

Opdracht 1 (max. 2.0 pnt)
Breid het laatste sql script uit met minimaal de volgende stored routines. Laat, indien van toepassing, ook stored routines andere stored routines aanroepen.

- sp_get_oligos_by_tm(min, max)
  geeft de oligos terug waarvan de temperaturen tussen min en max liggen.
- sp_get_matrices_by_quality
  geeft een lijst terug van alle aanwezige matrices oplopend geordend op aantal genen zonder probes, aflopend op aantal genen met maar één probe.
- sp_create_probe(matrix_id, oligo_id)
  zie naam.
- sp_remove_overlapping_probes_from_matrix(matrix_id)
  laat maar één probe van een serie overlappende probes in de matrix staan.
- sp_create_matrix(melting_t, max_difference)
  maakt in de database een nieuwe microarray op basis van de oligos in de database waarvan de smelttemperaturen van de probes binnen de melting_t +/- de max_difference liggen.

Opdracht 2 (max. 0.5 pnt)
Breid ook je python module uit zodat alle stored routines via methods in de klasse aan te roepen zijn. Items moeten als lists teruggegeven worden.