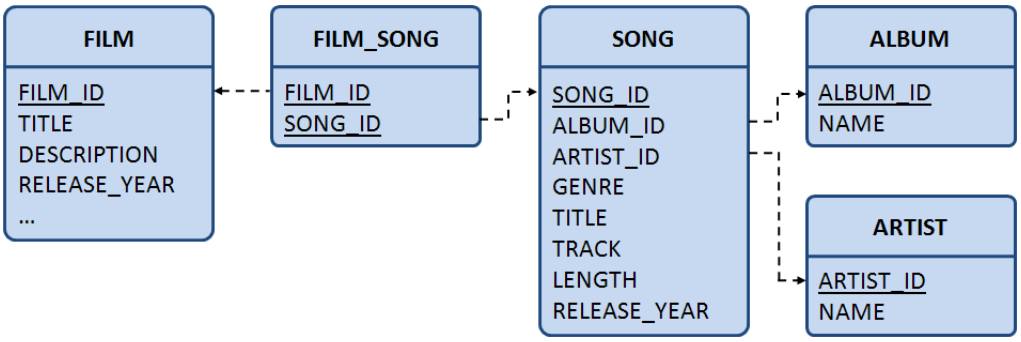
EXAMEN: **Scriptingtalen**

Prof. Dr. Peter Dawyndt
1e Bachelor Informatica
groep 3–4

maandag 23-06-2008, 14:00h
academiejaar 2007-2008
eerste zittijd

Opgave 1

De DVD-verhuurketen heeft zijn Sakila databank uitgebreid, waardoor informatie kan worden opgezocht over soundtracks van films. Hiervoor werden vier extra tabellen aan de oorspronkelijke databank toegevoegd. Onderstaande figuur geeft een overzicht van de nieuwe tabellen, waarbij (samengestelde) primaire sleutels onderlijnd zijn en vreemde sleutels worden aangegeven met onderbroken pijlen.



De tabel `SONG` houdt een lijst bij van liedjes, waarbij aan elk liedje een uniek rangnummer toegekend wordt (kolom `SONG_ID`). De kolom `ALBUM_ID` is een vreemde sleutel die verwijst naar de gelijknamige kolom van de tabel `ALBUM`, en die aangeeft op welk album het liedje voorkomt. De kolom `ARTIST_ID` is een vreemde sleutel die verwijst naar de gelijknamige kolom van de tabel `ARTIST`, en die aangeeft welke artiest het nummer geschreven heeft. De tabel `ALBUM` (resp. `ARTIST`) bevat verder enkel de naam van het album (resp. de artiest). Voorts geeft de tabel `SONG` ook nog informatie weer over de titel van het nummer (`TITLE`), het genre (`GENRE`), de plaats van het nummer op het album (`TRACK`), de lengte van het nummer (`LENGTH`) en het jaar waarin het werd uitgegeven (`RELEASE_YEAR`). De tabel `FILM_SONG` geeft aan welk nummer in welke film gebruikt wordt, en bevat daarvoor twee vreemde sleutels die verwijzen naar de primaire sleutels van de tabellen `FILM` en `SONG`.

De uitgebreide Sakila databank werd ingeladen in een MySQL RDBMS en is bevraagbaar via de phpMyAdmin webapplicatie (`http://users.ugent.be/~dhollevo/scrita_sql/`). Inloggen doe je in deze databank met je UGent account. Gevraagd wordt om SQL zoekopdrachten te formuleren die een antwoord bieden op onderstaande vragen. Zorg er telkens voor dat de kolommen van de resulterende tabel een zinvolle naam krijgen. Geef in je antwoordbestand ook aan hoeveel records de resultatentabel bevat.

1. Bereken het minimum, maximum en gemiddeld aantal liedjes per album.

2. Wat is het meest populair genre van alle liedjes die in de databank voorkomen. Geef ook aan hoeveel liedjes er tot dit genre behoren.

3. Geef een alfabetisch gerangschikte lijst van de titels van alle films waarin muziek gebruikt wordt van de artiest "The Prodigy". Let erop dat wegens spellingsinconsistenties, de naam van deze artiest in de volgende varianten voorkomt in de databank: "The Prodigy", "The prodigy" en "Prodigy".

4. Geef de namen van elk paar albums dat minstens twee liedjes bevat met dezelfde titel. Sorteer elk paar eerst op de eerste naam en dan op de tweede naam.

Opgave 2

Schrijf een macro `InitialiseerDeling` die een nieuw leeg werkblad aanmaakt in Excel, en die de eerste regel van dit werkblad opmaakt zoals aangegeven op onderstaande figuur.

	A	B	C	D	E	F	G	H
1		=		*		+		TRUE
2								
3								

Hierbij worden enkel de cellen $B1$, $D1$, $F1$ en $H1$ aangepast. In de eerste drie cellen komt telkens de aangegeven tekst, die gecentreerd in de cel geplaatst wordt. Cel $H1$ bevat een logische formule die aangeeft of de vergelijking $A1 = C1 * E1 + G1$ daadwerkelijk opgaat. Op deze cel is ook een voorwaardelijke opmaak van toepassing. Vult een gebruiker na het toepassen van deze macro $A1$, $C1$, $E1$ en $G1$ op met getallen, dan kleurt de achtergrond van cel $H1$ automatisch groen of rood naargelang de ingevulde bewerking klopt of niet.

	A	B	C	D	E	F	G	H
1	13	=	4	*	8	+	11	FALSE
2								
3								

Schrijf een tweede macro `StaartDeling` die de getallen uit de cellen $A1$ en $C1$ uitleest, en de staartdeling $A1/C1$ construeert compleet met opmaak zoals in onderstaand voorbeeld staat aangegeven.

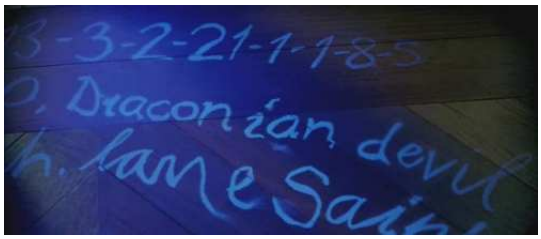
	A	B	C	D	E	F	G	H
1	123569	=	16	*	7723	+	1	TRUE
2								
3	1	2	3	5	6	9	16	
4	0						7723	
5	1							
6		12						
7		0						
8		12						
9			123					
10			-112					
11			11					
12				115				
13				-112				
14				3				
15					36			
16					-32			
17					4			
18						49		
19						-48		
20						1		
21								

Het quotiënt en de rest van de deling worden door de macro `StaartDeling` gekopieerd naar de corresponderende cellen op de eerste regel van het werkblad. Dankzij het voorbereidende werk van de macro `InitialiseerDeling` wordt dan ineens weergegeven of de deling correct werd uitgevoerd. Zorg er ook voor dat de resultaten kloppen voor deeltal gelijk aan nul en negatief deeltal en/of deler.

Hint: Het aantal gebruikte kolommen in een staartdeling is gelijk aan het aantal cijfers in het deeltal plus één (voor de deler). De eerste kolom moet altijd de kolom A zijn. Vergeet niet de vorige staartdeling te wissen alvorens een nieuwe te construeren.

Opgave 3

Een anagram is een woord dat of een zin die bestaat uit alle letters van een ander woord of een andere zin. Anagrammen worden vaak gebruikt als pseudoniem (*Leonardo da Vinci*: *o draconian devil*; *The Mona Lisa*: *oh lame saint*). Een eenvoudige manier om te bepalen of woorden anagrammen zijn, bestaat erin om een hash-waarde te berekenen zodat enkel anagrammen identieke waarden hebben. Voor de berekening van een hash-waarde kan je bijvoorbeeld aan elke letter van het alfabet een priemgetal toekennen (a=2, b=3, . . . , z=101), en de waarden die corresponderen met elke letter van het originele woord met elkaar vermenigvuldigen. Communtativiteit van de vermenigvuldiging en uniciteit van de ontbinding in priemfactoren garanderen dat anagrammen een unieke hash-waarde hebben. Bij de berekening van de hash-waarde moeten enkel de letters uit het alfabet in rekening gebracht worden (geen leestekens, cijfers, etc.), en moeten deze niet-hoofdlettergevoelig behandeld worden. Zo resulteren de woorden `callers`, `cellars` en `RECALLS` allemaal in de hash-waarde `615461330`, waardoor het dus anagrammen zijn.



Gevraagd wordt om een **awk**-script `anagram.awk` te schrijven dat alle combinaties van woorden die anagrammen zijn opzoekt in een gegeven woordenlijst. Het script leest de woordenlijst uit een bestand dat één enkel woord per regel bevat. Elke regel die door het script naar standaard uitvoer wordt weggeschreven moet alle woorden uit de woordenlijst bevatten die anagrammen van elkaar zijn. De anagrammen worden oplopend genummerd, waarbij voor elke index vier posities gereserveerd worden en de indices rechts worden uitgelijnd. Finaal moet het script ook *i*) het aantal anagrammen, *ii*) het totaal aantal woorden die een anagram hebben, *iii*) de langste woorden die anagrammen hebben en *iv*) de verzameling anagrammen die het meeste woorden bevat uitschrijven naar standaard uitvoer. Zorg ervoor dat het `awk`-script een functie `hash` bevat die voor een gegeven tekenreeks de hash-waarde berekent zoals hierboven werd beschreven. Voor de berekening van de hash-waarden moet het script eerst automatisch een lijst van de eerste 26 priemgetallen opstellen. Je kan hiervoor gebruik maken van een zelfgedefinieerde functie `ispriem` die nagaat of een gegeven getal n een priemgetal is door te controleren dat het niet deelbaar is door $2, 3, \ldots, \sqrt{n}$. Als we het script bijvoorbeeld uitvoeren op het voorbeeldbestand `dictionary.txt`, dan moet de volgende uitvoer gegenereerd worden.

```
    1: Kyoto,Tokyo
    2: budge,debug
...
   14: danger,gander,garden,ranged
...
```

```
2532: aims,Siam
2533: credits,directs
2534: mundane,unnamed

#anagrammen: 2534
#woorden: 5691
langste woord: algorithmically,logarithmically
meeste woorden: pares,parse,pears,rapes,reaps,spare,spear
```

Merk op dat de volgorde van de anagrammen niet noodzakelijk dezelfde hoeft te zijn als in bovenstaande uitvoer. Geef in je antwoordbestand ook de (ingekorte) uitvoer die gegenereerd wordt wanneer je je script uitvoert op het voorbeeldbestand `woordenboek.txt`.

Hint: Hou rekening met het feit dat `awk` bij automatische conversie van getallen naar tekenreeksen voor grote getallen steeds de wetenschappelijke notatie zal gebruiken. Zo wordt de hash-waarde van het woord `abbreviation` omgezet naar de tekenreeks `1.44855e+14` en niet naar 144854651795436. Je zal dus zelf moeten zorgen voor de gepaste conversie naar de laatste voorstelling.

Opgave 4

In de bioinformatica is BLAST (*Basic Local Alignment Search Tool*) een veelgebruikt algoritme voor het vergelijken van nucleotide- of eiwitsequenties. Met een BLAST zoekopdracht kunnen onderzoekers een onbekende sequentie (de zogenaamde *query*sequentie) zeer snel vergelijken met een databank van sequenties, en op die manier sequenties opzoeken die sterke gelijkenissen vertonen met de querysequentie. Wanneer een onderzoeker bijvoorbeeld een onbekend gen ontdekt in de muis, dan zal hij typisch een BLAST zoekopdracht uitvoeren tegenover het menselijk genoom om te kijken of de mens gelijkaardige genen heeft.

Het bestand `blast.out` bevat het resultaat van een BLAST zoekopdracht. Dit bestand bevat alle sequenties (de zogenaamde *subject*sequenties) uit de databank die een similariteitsscore vertonen met de querysequentie die boven een bepaalde drempelwaarde ligt. De basisinformatie van elke subjectsequentie begint met een regel die als eerste karakter een groter dan symbool (`>`) heeft. Deze informatie kan eventueel over meerdere regels lopen, en wordt gevolgd door een regel die de lengte van de subjectsequentie vermeldt en een lege regel. Hierna volgt een lijst van subsequenties die zeer sterk op elkaar gelijken in de querysequentie en de subjectsequentie, volgens dalende similariteitsscore. Voor elke subsequentie worden eerst twee regels met statistieken weergegeven: de eerste regel bevat oa. de similariteitsscore (`Score`) en de significantie van de vergelijking (`Expect`), en de tweede regel bevat oa. het aantal identieke posities (`Identities`) en het aantal positieve posities (`Positives`). Daarna volgt een gedetailleerd alignement tussen de querysequentie en de subjectsequentie, per drie regels gesplitst over meerdere regels met daartussen telkens een lege regel.

Gevraagd wordt:

1. Schrijf een `sed`-script `blast_list.sed` dat een overzicht van de subjectsequenties genereert voor een gegeven BLAST output. Elke regel in het overzicht bestaat uit zes informatievelden die elk een vaste breedte hebben, en waarbij de verschillende velden door één enkele spatie van elkaar worden gescheiden. De eerste twee velden bevatten de informatie uit de regel die start met het `>` symbool in de originele BLAST output, waarbij de eerste spatie beschouwd wordt als scheidingsteken tussen de twee velden. Beide velden worden in het overzicht links uitgelijnd. Het eerste veld is 20 karakters breed en overtollige informatie wordt gewoon afgekapt. Het tweede veld is 40 karakters breed. Indien de originele informatie voor het tweede veld langer is dan 37 karakters dan wordt het op die positie afgekapt en wordt de rest vervangen door drie puntjes.

De overige informatievelden bevatten de volgenden statistieken van de subsequentie met de hoogste similariteitsscore, waarbij de veldbreedte telkens tussen ronde haakjes staat: similariteitsscore (4), significantie van de vergelijking (8), procentueel aantal identieke posities (4) en procentueel aantal positieve posities (4). Deze velden worden allemaal rechts uitgelijnd.

De eerste vijf regels die door het commando `gsed -f blast_list.sed blast.out` gegenereerd worden, moeten er dan als volgt uitzien:

```
gnl|PID|e288614      (Z83122) R11A5.h [Caenorhabditis eleg...  173  1.3e-14  36%  79%
gi|1763275           (U73478) acidic nuclear phosphoprotei...  134  8.4e-11  33%  59%
gi|1769451           (X69465) ryanodine receptor 1 [Sus sc...  130  1.2e-09  46%  78%
gi|1752736           (D83006) gene required for phosphoyla...  121  8.5e-09  27%  66%
sp|P09741|TRT2_RABIT TROPONIN T, CARDIAC MUSCLE ISOFORMS       102  3.8e-08  29%  57%
```

2. Schrijf een `awk`-script `blast_extract.awk` dat informatie over één enkele subjectsequentie uit een BLAST output extraheert. De index van de subjectsequentie wordt aan de hand van de parameter `hit` aan het script doorgegeven. Indien er een ongeldige waarde voor deze parameter wordt opgegeven, moet het script een gepaste foutboodschap genereren. De informatie over de gevraagde subjectsequentie moet ongewijzigd naar standaard uitvoer worden geschreven, en wordt daarbij beperkt tot informatie over de subquery met de hoogste similariteitsscore. Zorg er ook voor dat de laatste regel die wordt uitgeschreven geen lege regel is. Zo genereert het commando `gawk -v hit=4 -f blast_extract.awk blast.out` de vierde subjectsequentie uit de BLAST output op de standaard uitvoer, wat het volgende resultaat geeft.

```
gi|1752736 (D83006) gene required for phosphoylation of oligosaccharides/ has
            high homology with YJR061w [Saccharomyces cerevisiae]
            Length = 1178

 Score = 121 (55.4 bits), Expect = 8.5e-09, P = 8.5e-09
 Identities = 23/84 (27%), Positives = 56/84 (66%)

Query:   139 DEEEEEEEEEEEGEEEEEEEEDDDDDDDEDSGAEIQDDDEEGFDDEEEFDDDEHDDDDLD 198
             +EEE++++EEEE +++EEEE+   +++++    E +   +E  + +++ + ++  ++D +
Sbjct:  1062 EEEEKKKKEEEEKKKKEEEEKKKKEEEEKKKQEEEEKKKKEEEEKKKQEEGEKMKNEDEE 1121

Query:   199 NEENELEELEERVEARKKTTEKQS 222
             N++NE EE ++  E  KK  E+++
Sbjct:  1122 NKKNEDEEKKKNEEEEKKKQEEKN 1145
```

3. Schrijf een `bash` shell script `blast_viewer.sh` dat bovenstaande scripts combineert tot een eenvoudig programma om BLAST output te bekijken. Aan dit script wordt een BLAST outputbestand als argument meegegeven. Na opstarten vraagt het script na de prompt `>>` naar gebruikerscommando's, die als volgt geïnterpreteerd worden:

   - `q` of `Q` sluit het programma af
   - `l [`n`]` of `L [`n`]` genereert een overzicht van de eerste n subjectsequenties uit de BLAST output, met als standaardwaarde n=5 indien deze parameter niet wordt opgegeven. Gebruik hiervoor natuurlijk het script `blast_list.sed`. Zorg ervoor dat de regels in het overzicht oplopend genummerd worden, zodat gebruikers makkelijk de index van een bepaalde subjectsequentie kunnen bepalen. Het is echter niet toegelaten om hiervoor het script `blast_list.sed` aan te passen.
   - alle andere invoer wordt beschouwd als parameter `hit` voor het script `blast_extract.awk`

Het script `blast_viewer.sh` moet nagaan of alle databestanden en scriptbestanden die het gebruikt bestaan en leesbaar zijn, en of er daadwerkelijk een argument aan het script wordt doorgegeven. Hieronder staat een voorbeeldsessie, waarbij het script `blast_viewer.sh` gebruikt wordt met de BLAST output `blast.out`.

```
bash$ blast_viewer.sh blast.out
>> L 10
  1. gnl|PID|e288614      (Z83122) R11A5.h [Caenorhabditis eleg...  173  1.3e-14  36%  79%
  2. gi|1763275           (U73478) acidic nuclear phosphoprotei...  134  8.4e-11  33%  59%
  3. gi|1769451           (X69465) ryanodine receptor 1 [Sus sc...  130  1.2e-09  46%  78%
  4. gi|1752736           (D83006) gene required for phosphoyla...  121  8.5e-09  27%  66%
  5. sp|P09741|TRT2_RABIT TROPONIN T, CARDIAC MUSCLE ISOFORMS       102  3.8e-08  29%  57%
  6. gi|1754694           (U81159) magnesium-dependent calcium ...  109  2.1e-06  36%  67%
  7. sp|Q08168|HRP_PLABE  58 KD PHOSPHOPROTEIN (HEAT SHOCK-RELA...   98  5.1e-06  66%  81%
  8. sp|Q10713|MPP1_HUMAN MITOCHONDRIAL PROCESSING PEPTIDASE AL...   94  4.2e-05  22%  44%
  9. gi|1762998           (U67260) RING-finger protein [Helicov...   92  7.4e-05  21%  47%
 10. sp|P02547|NFL_PIG    NEUROFILAMENT TRIPLET L PROTEIN (68 K...   93  0.00039  32%  57%
>> 1
gnl|PID|e288614 (Z83122) R11A5.h [Caenorhabditis elegans]
            Length = 906

 Score = 173 (79.1 bits), Expect = 1.3e-14, Sum P(2) = 1.3e-14
 Identities = 27/73 (36%), Positives = 58/73 (79%)

Query:   131 EGKTGNATDEEEEEEEEEEGEEEEEEEDDDDDDDEDSGAEIQDDDEEGFDDEEEFDDD 190
             E K  ++ +EE++++E+EEE E+EEEE+ED++D+++E++ +E +++D+E  + EEE D++
Sbjct:   576 ELKLDDSDEEEDDDDEDEEEDEDEEEEDEDEEDEEEEENESEEEEEDDEDEESEEESDEE 635

Query:   191 EHDDDDLDNEENE 203
             + ++++ D+ E E
Sbjct:   636 DEEEEEEDDSEPE 648
>> q
bash$
```