# CS3 Database Systems

Coursework 1. Due Monday, 18 October, 2008

*Please staple your answers and put your NAME and STUDENT ID on the front. Hand them into the ITO no later than 4pm on Friday, 18 October.*

All questions require short answers.

1. Suppose you have to collect data from a large system of weather stations. Each station has an ID. It records TEMPerature, PRESsure, HUMidity, and WINDspeed. Each station has a position consisting of a LATitude, LONGitude and ELEVation. Stations come in two kinds: STATic and MOBile. In mobile stations (such as balloons and ships) the position varies.

   PRECipitation (rainfall) is recorded only by static stations. They do this by using a little bucket which fills up each time 1mm of rain accumulates, sends a signal, and then empties itself. Thus preciptation is recorded as a series of times.

   Weather stations normally take readings at approximately 15 minute intervals. In addition to the data described above, each reading contains its time and the preciptation represented as a list of times since the last reading. Readings are normally sent from a station to the central office, but since contact with the office is often lost, stations buffer the data and send it as a file containing a list of readings when contact is re-established.

   (a) Design a DTD for the file. The DTD should be reasonably "modular". E.g., there should be a *position* element. The same DTD should be used by both static and moving stations.

   *Answer*:

   ⟨!ELEMENT READING-LIST (ID, (STAT | MOB))⟩
   *Each station sends its ID and a STATic or MOBile list of readings*
   ⟨!ELEMENT STAT (POS, STATLIST)⟩
   *If it's static the position fixed*
   ⟨!ELEMENT POS (LAT, LONG, ELEV)⟩
   *The position*
   ⟨!ELEMENT STATLIST (STATREADING*)⟩
   *List of static readings*
   ⟨!ELEMENT STATREADING (TIME, TEMP, PRES, HUM, RAINFALL)⟩
   *Static readings record rainfall*
   ⟨!ELEMENT TIME (#PCDATA)⟩
   *No types. Every basic type is a character string.*
   ⟨!ELEMENT TEMP (#PCDATA)⟩
   ⟨!ELEMENT PRES (#PCDATA)⟩
   ⟨!ELEMENT HUM (#PCDATA)⟩
   ⟨!ELEMENT RAINFALL (TIME*)⟩
   *A list of times. Could instead have recorded count of these*
   ⟨!ELEMENT MOB, (MOBREADING*)⟩

*Mobile readings are different from static ones*
⟨!ELEMENT MOBREADING (TIME, POS, TEMP, PRES, HUM)⟩
*They record positions but not rainfall*

Notes: There are all sorts of acceptable variations. There are some simple substitutions that could be made (perhaps at the expense of some clarity) such as ⟨!ELEMENT STAT (POS, STATREADING*)⟩. This DTD tries to minimise the redundancy. For example POS is not placed inside a static "reading", but it could be argued that if multiple readings are rare, then it doesn't matter if they are repeated.

(b) The central office also provides historical data in which the readings are ordered by time; for example you can request an ordered list of all the readings in a 24 hour period. Design a DTD for the data they will send you. Be sure to "recycle" elements from your previous answer where possible.

*Answer*:

⟨!ELEMENT READINGLIST (START, END, READING*)⟩
⟨!ELEMENT START (#PCDATA)⟩
⟨!ELEMENT END (#PCDATA)⟩
*The start and end times*
⟨!ELEMENT READING (TIME,ID, POS, TEMP, PRES, HUM, RAINFALL?)⟩
*Readings ordered by time*
Other definitions as before

There is some redundancy in this. The POSition is fixed for static stations bit repeated in this representation. It's not clear how to avoid this. One could also have an explicit flag for static/dynamic stations

Please note that their is no single "best" soultion to this problem. If you are in doubt and make assumptions, be sure to state them clearly.

2. Consider the following DTD (all undefined elements are character data) :

⟨!ELEMENT ROOT (B|C*)⟩
⟨!ELEMENT B (G,D)⟩
⟨!ELEMENT D B*⟩
⟨!ELEMENT C (D,E)⟩
⟨!ELEMENT E (D,C)⟩

Each of the following XPATH expressions can be simplified if we know that the document satisfies this DTD. In each case give at least one simplification.

(a) (G|D|H) at a context node with tag B.
*Answer*: (G|D)

(b) //B//G
*Answer*: B/G or //G

(c) //B//D//B
*Answer*: /B//B

(d) //E

*Answer*: [1=2] – however you say "nothing"!

3. Consider the relation scheme $R(\underline{A}, B), R'(\underline{A}, B), S(\underline{B}, C), T(\underline{C}, B)$ In each of the following statements of the form "$k$ is a key for $e$" state whether it is true or false. If false, give an example of relevant tables.

   (a) $A$ is a key for $R \setminus R'$.
   *Answer*: True

   (b) $A$ is a key for $R \cup R'$
   *Answer*: False. $R = \{(1,2)\}, R' = \{(1,2)\}$

   (c) $B$ is a key for $R \bowtie S$.
   *Answer*: False. $R = \{(1,2),(3,2)\}, B = \{2,5\}$.

   (d) $B$ is a key for $R \bowtie S \bowtie T$.
   *Answer*: False. $R = \{(1,3),(2,3)\}, S = \{(3,4)\}, T = \{(4,3)\}$

   (e) $A$ is a key for $R \cup (R' \setminus \pi_{A,B}(R' \bowtie \pi_C(S)))$
   *Answer*: False. If $S = \{\}$ then we are asking if $A$ is a key for $R \cup R'$

4. Each of the following relational algebra expressions can be rewritten to an expression that is probably more efficient. In each case give such an expression. Assume that the relation schemes are such that all expressions make sense.

   (a) $\sigma_{A=5}(R) \cup \sigma_{B=6}(R)$
   *Answer*: $\sigma_{A=5 \vee B=6}(R)$

   (b) $\sigma_{A<B}(R) \cap \sigma_{B=C}(R) \cap \sigma_{C>A}(R)$
   *Answer*: $\sigma_{A<B \wedge B=C}(R)$

   (c) $\sigma_{A=5}(\sigma_{A>2}(R) \bowtie S)$
   *Answer*: $\sigma_{A=5}(R) \bowtie S$

   (d) $\pi_{AB}(R) \bowtie \pi_{AC}(R)$ when $A$ is a key for $R$
   *Answer*: $\pi_{ABC}(R)$

   (e) $(\pi_{AB}(R) \bowtie \pi_{CD}(R)) \bowtie \pi_{BD}(R)$
   *Answer*: $(\pi_{AB}(R) \bowtie \pi_{BD}(R)) \bowtie \pi_{CD}(R)$

5. Given a relation $R(A, B, \ldots)$, write a relational algebra expression that determines whether or not $R$ satisfies the constraint that $A$ is a key. Your expression should yield a non-empty table if the key constraint is violated and an empty table otherwise.

   *Answer*: Consider the table $T = R \bowtie \rho_{B \to B', C \to C', \ldots}(R)$. If $A$ is a key then the $B$ and $B'$, the $C$ and $C'$, etc. values of each tuple will be the same. Thus $\sigma_{B \neq B' \vee C \neq C' \ldots}(T)$ will be non-empty if the key constraint is violated.

6. Using the Climbs table of the Munros/Hikers database in class, use relational algebra find to the names of each hiker who has the fastest time on

(a) all the mountains that the hiker has climbed, and

*Answer*: Use $C(H, M, D, T)$ as an abbreviation.
Let $R_1 = C \bowtie \rho_{H \to H', T \to T'}(\pi_{HMT}(C))$.
If there is a tuple with $T < T'$ and $H \neq H'$ then hiker $H$ will have been beaten by someone (Id $H'$) on that mountain. So
$R_2 = \pi_H(\sigma_{T > T' \wedge H \neq H'}(R_1))$
gives us all the climbers who have been beaten on some mountain.
$R_3 = \pi_H(\mathsf{Hikers}) \setminus R_2$
is now the set of IDs of hikers who have not been beaten. Note that a hiker can satisfy this query by not climbing anything! Now do a join to get the names.

(b) some mountain that the hiker has climbed

*Answer*: Compute $R_1$ as before. Now let
$R_4 = \pi_{HM}(\sigma_{T > T' \wedge H \neq H'}(R_1))$
This is a table of hiker/mountain pairs for which the hiker is a "loser" for that mountain. If we now compute
$R_5 = \pi_{HM}(C) \setminus R_4$
we get each mountain with the "winners" for that mountain. $\pi_H(R_5)$ will be the people who have won for some mountain and we can join with the Hikers table to get their names.

7. Provide a relational schema for the data held at the central office in question 1. Assume types such as string, integer, float, date etc. and be sure to state the keys for your table. *Answer*: Using SQL DDL

```
CREATE TABLE STATICSTATIONS(
    ID INT,
    LAT FLOAT,
    LONG FLOAT,
    HEIGHT FLOAT,
PRIMARY KEY ID)

CREATE TABLE STATICREADING(
    ID INT,
    TIME TIME,
    TEMP FLOAT,
    PRES FLOAT,
    HUM FLOAT,
    PRIMARY KEY (ID, TIME),
    FOREIGN KEY ID REFERENCES STATICSATIONS)

CREATE TABLE RAINFALL(
    ID INT,
    BUCKETDIP TIME, – The time at which the bucket filled
    PRIMARY KEY (ID, BUCKETDIP)
    FOREIGN KEY ID REFERENCES STATICSTATIONS)
```

*Alternatively one could simply record the rainfall – number of dips – in STATICREADING*

```
CREATE TABLE MOBILEREADING(
   ID INT,
   TIME TIME,
   LAT FLOAT,
   LONG FLOAT,
   HEIGHT FLOAT,
   TEMP FLOAT,
   PRES FLOAT,
   HUM FLOAT
   PRIMARY KEY(ID,TIME) )
```

8. Assuming there are 1000 weather stations in the UK and, as indicated in question 1, they report a reading every 15 minutes, estimate the total amount of data that the central office will need to store every year. Average rainfall in the UK is about 1 metre. *Answer*: Let's work in bytes and assume all stations static, and use the schema above. Table STATICSTATIONS: $1000 \times (4 + 3 \times 8) = 28$KB (4 bytes for INT and 8 bytes for float). Table STATICREADING: $1000 \times 365 \times 24 \times 4 \times (4 + 8 + 8 + 8 + 8) = 1261$GB (8 bytes for time). Table RAINFALL $1000 \times 1000 \times (8 + 8) = 16$MB (Each station experiences 1000 bucket dips on average)

   The 1261GB figure dominates (double it to get the store needed for a relational DB). Incidentally, it appears to be more space-efficient to store the times of the bucket dips rather putting the accumulated rainfall in the STATICREADINGS!