UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER SCIENCE THREE

DATABASE SYSTEMS

Friday 31 May 2002

09:30 to 11:00

Examiners:

Prof. K. Turner (External)

Prof. C. Stirling (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. A company payroll database contains information about people who are paid (payees), projects, and what projects people work on. There are two kinds of payee: employee and consultant. Here is a relational schema for the database.

$$
\begin{array}{l}
\textit{project}(\underline{\textit{pId}}, \textit{pName}, \textit{budget}) \\
\textit{payee}(\underline{\textit{id}}, \textit{name}) \\
\textit{employee}(\underline{\textit{id}}, \textit{salary}, \textit{office}) \\
\textit{consultant}(\underline{\textit{id}}, \textit{hourly\_rate}, \textit{address}) \\
\textit{works\_on}(\underline{\textit{id}}, \underline{\textit{pId}}, \textit{hours\_worked})
\end{array}
$$

   (a) Draw an E-R diagram that represents the database and captures the information above. Underline the attributes in your diagram that are keys or parts of keys. [5 *points*]

   (b) The *payee* entity set can be translated into the SQL DDL

```
CREATE TABLE payee ( id   CHAR(9),
                     name CHAR(40),
                     PRIMARY KEY (id) )
```

   Similarly translate the *works_on* and *employee* tables, specifying foreign keys [the syntax for foreign key is `FOREIGN KEY (...)  REFERENCES` R`(...)`]. Use whatever types you think are appropriate for domains. [6 *points*]

   (c) Write SQL queries for the following.

      i. The names of *employees* who worked on the "Widget" project. [6 *points*]

      ii. The names of *payees* who have worked on all projects. [8 *points*]

2. (a) For each of the following, state whether it is true or false and briefly justify your answer.

    i. Starting from the same initial state, any two serialisable schedules will leave the database in the same state. [2 *points*]

    ii. Two-phase locking guarantees serialisability. [2 *points*]

    iii. Two-phase locking guarantees no deadlock. [2 *points*]

   (b) Consider the following transactions

T_1:	T_2:	T_3:
read (X); $X := X + 3$; write (X)	read (X); $X := X + 5$; write (X)	read (X); $X := X * 2$; write (X)

    i. Given that the initial value of X is 0, write down all possible values of X that result from serialisable executions of T_1, T_2 and T_3. [4 *points*]

    ii. Write down a non-serialisable schedule of T_1, T_2 and T_3. [3 *points*]

    iii. What is the value of X that results from this schedule? [2 *points*]

    iv. Draw the conflict graph for this schedule. [3 *points*]

   (c) Given the functional dependency set

$$\{A \to B, B \to CD, C \to AE, DE \to A, EF \to G\}$$

    i. Give a candidate key for $ABCDEFG$ (a candidate key is a minimal set of attributes that determines $ABCDEFG$). [4 *points*]

    ii. Is your answer to 2(c)i unique? If not, give another candidate key. [3 *points*]

3. (a) Briefly distinguish between the terms *well-formed* and *valid* when applied to an XML document. [3 *points*]

(b) Consider the following DTD:

```
<!ELEMENT db (r,s)>
<!ELEMENT r (rt*)>
<!ELEMENT s (st*)>
<!ELEMENT rt (a,b,c)>
<!ELEMENT st (d,e,f)>
...
```

All undefined tags are `#PCDATA`.

i. Write down an appropriate SQL DDL to store XML data conforming to this DTD. Assume `CHAR(10)` for `#PCDATA`. [4 *points*]

ii. What property or properties of the XML data would your relational implementation fail to capture? [3 *points*]

(c) The following assertions, of the form $p_1 \subseteq p_2$, state that, for *any* XML document and *any* context node, the set of nodes identified by XPath expression p_1 is included in the set of nodes identified by p_2. For each assertion, say whether it is true. If it is false, and only then, give an example of an XML document and context node (you may draw a tree diagram) for which p_1 identifies a node that p_2 does not.

i. `aaa` $\subseteq$ `(aaa|bbb)` [2 *points*]

ii. `aaa` $\subseteq$ `aaa/bbb` [2 *points*]

iii. `aaa/bbb` $\subseteq$ `aaa//bbb` [2 *points*]

iv. `aaa` $\subseteq$ `//aaa` [2 *points*]

v. `.//aaa` $\subseteq$ `aaa//aaa` [2 *points*]

(d) Given that the file `xml.xml` contains the xml document

```
  <a>
    <b> <c> C1 </c> <c> C2 </c> <d> D1 </d> </b>
    <b> <c> C3 </c> <d> D2 </d> </b>
 </a>
```

What does the following XQuery query produce?

```
<answer>{
   for $x in document("xml.xml")
     for $y in $x/b/c
     return <b> {$y} <e> {$x/b/d} </e> </b>
}</answer>
```

[5 *points*]