CSCE 520 – Section 002

Homework 6

5.3.1 (a, b, c) – Write each of the queries of Exercise 2.4.1 in Datalog.

1. What PC models have a speed of at least 3.00?

H(model) ← PC(model, speed) AND speed ≥ 3.00

1. Which manufacturers make laptops with a hard disk size of at least 100 GB?

H(maker) ← Product(maker, model) AND Laptops(model, hd) AND hd ≥ 100

1. Find the model number and price of all products (of any type) made by manufacturer B.

H(model, price) ← PC(model, price) AND

Laptop(model, price) AND

Printer(model, price) AND

Product(maker, model) AND maker = ‘B’

6.6.1 (a, b) –

1. Given a speed and amount of RAM, look up the PC’s with that speed and RAM, printing the model number and price of each.

SET TRANSACTION READ ONLY;

BEING TRANSACTION;

SELECT model AND price

FROM PC

WHERE PC.speed = speed AND PC.ram = ram;

COMMIT; //end successfully even though no changes made to DB?

1. Given a model number, delete tuple for that model in both PC and Product.

BEGIN TRANSACTION;

DELETE FROM Product AND PC

WHERE Product.model = model AND PC.model = model;

COMMIT;

11.1.2 – Suggest how typical data about banks and customers, as in Exercise 4.1.1, could be represented in the semi structured model.

When it comes to data about banks and their customers, the amount of data being collected by each customer can vary greatly. One customer can only need a checking account, while another customer could require a checking, savings, business, and family accounts. Building a structured database being able to hold all this information for each customer can be wasteful and inefficient. By using a semi structured model, the customers can be grouped into more specific databases based on their needs while still being able to create an intuitive UI that would still allow access to all the account data that specific customer would need to see.

11.2.2 – Show that any relation can be represented by an XML document.

Student(name, id)

Class(prefix, dept)

<? Xml version = “1.0” encoding = “utf-8” standalone = “yes”?>

<StudentClassData>

<Student>

<Name>George Gordon </Name>

<Id>00219254</Id>

<Class>

<Prefix>520</Prefix>

<Dept>CSCE</Dept>

</Class>

</Student>

</StudentClassData>

11.3.1 (a, b) –

1. Carrie Fisher and Mark Hamill also starred in The Empire Strikes Back (1980) and Return of the Jedi (1983).

<? Xml version = “1.0” encoding = “utf-8” standalone = “yes”?>

<StarMovieData>

<Star starId = “cf” starredIn=”sw esb rotj”>

<Name>Carrie Fisher</Name>

<Address>

<Street>123 Maple St.</Street>

<City>Hollywood</City>

</Address>

<Address>

<Street>5 Loctus Ln.</Street>

<City>Malibu</City>

</Address>

</Star>

<Star starId = “mh” starredIn=”sw esb rotj”>

<Name>Mark Hamill</Name>

<Address>

<Street>456 Oak Rd.</Street>

<City>Brentwood</City>

</Address>

</Star>

<Movie movieId = “sw” starsOf = “cf mh”>

<Title>Star Wars</Title>

<Year>1977</Year>

</Movie>

<Movie movieId = “esb” starsOf = “cf mh”>

<Title>The Empire Strikes Back</Title>

<Year>1980</Year>

</Movie>

<Movie movieId = “rotj” starsOf = “cf mh”>

<Title>Return of the Jedi</Title>

<Year>1983</Year>

</Movie>

</StudentClassData>

1. Harrison Ford also starred in Star Wars, in the two movies mentioned in (a), and the movie Firewall (2006).

<? Xml version = “1.0” encoding = “utf-8” standalone = “yes”?>

<StarMovieData>

<Star starId = “cf” starredIn=”sw esb rotj”>

<Name>Carrie Fisher</Name>

<Address>

<Street>123 Maple St.</Street>

<City>Hollywood</City>

</Address>

<Address>

<Street>5 Loctus Ln.</Street>

<City>Malibu</City>

</Address>

</Star>

<Star starId = “mh” starredIn=”sw esb rotj”>

<Name>Mark Hamill</Name>

<Address>

<Street>456 Oak Rd.</Street>

<City>Brentwood</City>

</Address>

</Star>

<Star starId = “hf” starredIn=”esb rotj fire”>

<Name>Harrison Ford</Name>

<Address>

<Street>123 Somewhere Rd.</Street>

<City>Hollywood</City>

</Address>

</Star>

<Movie movieId = “sw” starsOf = “cf mh”>

<Title>Star Wars</Title>

<Year>1977</Year>

</Movie>

<Movie movieId = “esb” starsOf = “cf mh”>

<Title>The Empire Strikes Back</Title>

<Year>1980</Year>

</Movie>

<Movie movieId = “rotj” starsOf = “cf mh”>

<Title>Return of the Jedi</Title>

<Year>1983</Year>

</Movie>

<Movie movieId = “fire” starsOf = “hf”>

<Title>Firewall</Title>

<Year>2006</Year>

</Movie>

</StudentClassData>

11.3.2 – Suggest how typical data about banks and customers, as was described in Exercise 4.1.1, could be represented as a DTD.

In a DTD, all the attributes needed to infer references are given ID’s. These allow relations to be made very easily at the root level. This would be useful for banks when customers can be linked to many accounts at the same time. The relation between the account and customer will be well defined and distinguishable.

12.1.2 (b, d) – XPath queries.

b) Find all the Class elements for classes with a displacement larger than 35000.

/Ships/Class[//displacement > 35000]/@\*

d) Find the names of the ships that were sunk.

/Ships/Class/Ship/Battle[//outcome = “sunk”]/name

12.2.1 (b) – Write XQuery.

(b) Find the Printer elements with a price less than 100, and produce the sequence of these elements surrounded by a tag <CheapPrinters>.

let $printerSeq := (

let $products := doc(“products.xml”)

for $printer in $products/Products/Maker/Printer

where $printer/@price < 100

return $printer/@\*

)

return <CheapPrinter>{$printerSeq}</CheapPrinter>