CSE 344
2013Wi
Section 2 (1/17) Worksheet
Joins, Aggregates, and Indices

Question 1: Different Types of Joins

Let's say I have the following create table statements:

```
CREATE TABLE PokemonNames(
Id INTEGER,
Name VARCHAR(30),
PRIMARY KEY (Id));

CREATE TABLE PokemonTypes(
Id INTEGER REFERENCES PokemonNames,
Type Varchar(30));

INSERT INTO PokemonNames VALUES(7, "Squirtle");
INSERT INTO PokemonNames VALUES(65, "Alakazam");
INSERT INTO PokemonNames VALUES(129, "Magikarp");
INSERT INTO PokemonNames VALUES(147, "Dratini");
INSERT INTO PokemonNames VALUES(NULL, NULL);
INSERT INTO PokemonNames VALUES(NULL, NULL);

INSERT INTO PokemonTypes VALUES(65, "Psychic");
INSERT INTO PokemonTypes VALUES(NULL, NULL);
INSERT INTO PokemonTypes VALUES(147, "Dragon");
INSERT INTO PokemonTypes VALUES(NULL, NULL);
INSERT INTO PokemonTypes VALUES(7, "Water");
INSERT INTO PokemonTypes VALUES(129, "Water");
```

And the tables have been populated as such:

PokemonNames

Id	Name
7	Squirtle
65	Alakazam
129	Magikarp
147	Dratini
148	null
149	null

PokemonTypes

Id	Type
65	Psychic
null	null
147	Dragon
null	null
7	Water
129	Water

Please list the resulting tuples from the following queries that exercise different types of joins:

A) SELECT *
FROM PokemonNames n, PokemonTypes t
WHERE n.Id = t.Id

ANSWER:

Id	*Name*	*Id*	*Type*
65	*Alakazam*	*65*	*Psychic*
147	*Dratini*	*147*	*Dragon*
7	*Squirtle*	*7*	*Water*
129	*Magikarp*	*129*	*Water*

B) SELECT *
FROM PokemonNames n
LEFT OUTER JOIN PokemonTypes t ON n.Id = t.Id

ANSWER

Id	*Name*	*Id*	*Type*
7	*Squirtle*	*7*	*Water*
65	*Alakazam*	*65*	*Psychic*
129	*Magikarp*	*129*	*Water*
147	*Dratini*	*147*	*Dragon*
148	*NULL*	*NULL*	*NULL*
149	*NULL*	*NULL*	*NULL*

C) SELECT *
FROM PokemonNames n
RIGHT OUTER JOIN PokemonTypes t ON n.Id = t.Id

ANSWER

Id	*Name*	*Id*	*Type*
7	*Squirtle*	*7*	*Water*
65	*Alakazam*	*65*	*Psychic*
129	*Magikarp*	*129*	*Water*
147	*Dratini*	*147*	*Dragon*
NULL	*NULL*	*NULL*	*NULL*
NULL	*NULL*	*NULL*	*NULL*

Question 2: BASIC AGGREGATES

A) Easy

Write a SQL query for the above Pokemon example data that finds the number of Pokemon for each type. The resulting tuples should look something like this:

Type	TypeCount
NULL	2
Dragon	1
Psychic	1
Water	2

SOLUTION:
SELECT Type, Count(*) as TypeCount
FROM PokemonTypes
GROUP BY Type;

B) Harder (this is a midterm level problem)

A database with the following schema stores a collection of Webpages and the words they contain, and a collection of dictionaries in several languages and the words in those languages:

Occurs(url, word)
Dictionary(language, word)

- url represents a Webpage.
- Every Webpage may contain several words, and every word may occur in several Webpages.
- Every language may contain several words, and every word may occur in several languages
- There are no nulls in the database.

Write a SQL query that retrieves all languages that occur in more than 1000 Webpages. A language "occurs" in a Webpage if the Webpage contains a word in that language.

SOLUTION (11sp Midterm):
select y.language
from Occurs x, Dictionary y
where x.word = y.word
group by y.language

having count(distinct url) > 1000

C) Hard (another midterm level problem)

You are in charge of managing the program committee for an important conference. The following database stores information about papers submitted to the conference (table Paper), reviewers on the program committee (table Reviewer), and the assignment of reviewers to papers (table Reviews). Each reviewer on the program committee will have to review a set of papers. Each paper will be reviewed by some subset of reviewers.

Paper(pid, title)
Reviewer(rid, name)
Reviews(rid, pid)
- pid is a unique paper identier and the primary key of the Paper table.
- rid is a unique reviewer identier and the primary key of the Reviewer table.
- Reviews.rid is a foreign key that references Reviewer.rid.
- Reviews.pid is a foreign key that references Paper.pid.
- A reviewer is assigned zero or more papers.
- A paper is assigned zero or more reviewers.

Write a SQL query that finds all papers with fewer than three reviewers assigned to them. The output of the query should be a list of paper titles. The result should include papers without any reviewers assigned to them.

(HINT: Pay close attention to which the type of join you use)

SOLUTION:

```
SELECT P.title
FROM Paper P LEFT OUTER JOIN Reviews X ON P.pid = X.pid
GROUP BY P.pid, P.title
HAVING count(*) < 3
```