# Hw2

Joshua Klein

10/13/16

58485794

Problems 6.10(c)(d), 6.12, 6.13, 6.14

6.10 Write the following queries in relational algebra, using the university

schema.

a. Find the names of all students who have taken at least one Comp.Sci.

course. **π student.name ((student ⨝ takes) ⨝ (σ course.dept\_name = "CompSci" (course)))**

b. Find the IDs and names of all students who have not taken any course

offering before Spring 2009. **π student.name, student.ID (σ takes.semester = "spring" ∧ takes.year = 2009 ∨ takes.year < 2009 ( student ⨝ takes ⨝ course))**

c. For each department, ﬁnd the maximum salary of instructors in that

department. You may assume that every department has at least one

instructor. **γ dept\_name; max(salary) -> x (instructor)**

d. Find the lowest, across all departments, of the per-department maxi-mum salary computed by the preceding query. **γ min(x) -> new (γ dept\_name; max(salary) -> x (instructor))**

**6.12** Using the university example, write relational-algebra queries to ﬁnd the

Course sections taught by more than one instructor in the following ways:

a. Using an aggregate function. **π teaches.course\_id σ x > 1 γ course\_id ; count(course\_id) -> x ((instructor) ⨝ (teaches))**

b. Without using any aggregate functions. **π instructor.name ((π ID (σ teaches.ID = teaches2.ID (teaches ⨯ ρ teaches2 (teaches)))) ⨝ (π instructor.ID, instructor.name (instructor)))**

**6.13** Consider the relational database of Figure 6.22. Give a relational-algebra

Expression for each of the following queries:

a. Find the company with the most employees. **π company.name, number\_of\_emplyees (γ max(number\_of\_employees)->number\_of\_employees (γ count(company.company\_name)-> number\_of\_employees (works ⨝ company)))**

b. Find the company with the smallest payroll. **π company.name, number\_of\_emplyees (γ min(payroll)->payroll (γ sum(works.salary)-> payroll (works ⨝ company)))**

c. Find those companies whose employees earn a higher salary, on average, than the average salary at First Bank Corporation. **π x.company\_name σ x.company\_name != 'First Bank Corporation' ∧ y.company\_name = 'First Bank Corporation' ∧ x.average\_salary > y.average\_salary ⁃ (ρ x (γ company\_name; average(salary) -> average\_salary (works)) ⨯ ρ y (γ company\_name; average(salary) -> average\_salary (works)))**

**6.14** Consider the following relational schema for a library: member(memb no, name, dob) books (isbn, title, authors, publisher)borrowed(memb no, isbn, date) Write the following queries in relational algebra.

a. Find the names of members who have borrowed any book published by “McGraw-Hill”. **π member.name σ books.publisher = 'McGraw-Hill' (books ⨝ (member ⨝ borrowed))**

b. Find the name of members who have borrowed all books published by “McGraw-Hill”.

**π y.name σ x.book\_count = y.mem\_count (ρ x (γ books.publisher; count(publisher) -> book\_count (books))**

c. Find the name and membership number of members who have borrowed more than ﬁve different books published by “McGraw-Hill”. **Π(member.name(γ borrowed\_all\_books(borrowed.isbn)(member ⨝ books ⨝ borrowed)))**

d. For each publisher, ﬁnd the name and membership number of members who have borrowed more than ﬁve books of that publisher. **πmember\_n, number\_of\_publishers( γ count(books.publisher)-> number\_of\_publishers, first(member.name)->member\_n (member ⨝ books ⨝ borrowed))**

e. Find the average number of books borrowed per member. Take into account that if an member does not borrow any books, then that member does not appear in the borrowed relation at all. **γ avg(number\_borrowed)->borrowed (γ count(borrowed)-> number\_borrowed, member (member ⨝ books ⨝ borrowed))**