THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

# 7b. HW/Exam Review

## CSCI 2541 Database Systems & Team Projects

Wood & Chaufournier

# Today...

- Exam Logistics
- Relational Algebra
- SQL HW Review
- Normalization HW Review
- Lab: Work on your shopping cart HW
  - Ask us lots of questions!

# Exam Logistics

## Monday March 1st starting at 12:45PM

- You should join Zoom class as normal (~5 minutes early)
- Keep yourself muted, use "Raise Hand" for any questions
- Stay in Zoom until you finish the test

## Exam will be on Blackboard

- Short answer, multiple choice, T/F, etc
- Expect questions similar to Homeworks

## Class ends at 2PM

- I will give everyone 10 extra minutes to account for connectivity / tech issues

# You…

SELECT *

may:

- Use your own notes, my slides, your homework, SQL zine, Simha's notes, pages **directly** linked from my website
- Ask us questions

Resources Page

may not:

- Do random google searches
- Discuss questions or get help from anyone else
- Re-watch course videos
- Do anything else which violates the course or GW's academic integrity policies

Violating these policies will have severe consequences, including **failing** the course

# Suggestions

Make your own notes

- There are too many slides to search through during the exam
- Pick out pages from my PDFs or write up your own with references to the slides
- Writing out your own version of the notes (2NF vs 3NF, lossless decomposition rules, etc) will help you fully understand them!

Be an efficient test taker

- Hopefully nobody will get 100% on the exam
- Focus first on the sections you are most confident with
- Don’t waste too much time on any one question

# Schema for Bank database:

Customer (CustID, Name, street, city, zip)

- Customer ID, Name, and Address info: street, city, zip

Deposit (CustID, Acct-num, balance, Branch-name)

- Customer ID, Account number, Balance in account, name of branch where account is held;

CustID is foreign key referencing Customer.

- Branch-name is foreign key referencing Branch relation

Loan (CustID, Loan-num, Amount, Branch-name)

- Customer ID, loan number, amount of loan; CustID is foreign key referencing Customer relation;
- Branch-name is foreign key referencing Branch relation.

Branch (Branch-name, assets, Branch-city)

- Name of the branch (unique name), assets in dollars, and the city where the branch is located.

# Relational Algebra

**Loan** (CustID, Loan-num, Amount, Branch-name)

(1) Find tuples in loan where the amount is greater than 1300 and made in the “Downtown” branch.

$$\sigma_{amount > 1300 \wedge branch = DTown} \, LOAN$$

(2) Find names of customers whose name is the same as the street they live on.

**Customer** (CustID, Name, street, city, zip)

$$\Pi_{name} \, \sigma_{name = street} \, Customer$$

# Relational Algebra

Find tuples in loan where the amount is greater than 1300 and made in the “Downtown” branch.

$$\sigma_{amount>1300 \wedge branch-name="Downtown"}(Loan)$$

Find names of customers whose name is the same as the street they live on.

$$\Pi_{name}\, \sigma_{name=street}(Customer)$$

# Relational Algebra 2

Find assets and name of all branches that have depositors (i.e., customers with a Deposit account) living in New York.

**Customer** (CustID, Name, street, city, zip)

**Deposit** (CustID, Acct-num, balance, Branch-name)

**Branch** (Branch-name, assets, Branch-city)

$$CustNY \leftarrow \sigma_{cust.city='NY'} Cust$$

$$DepositsNY \leftarrow CustNY \bowtie Deposit$$

$$\Pi_{assets,\ branch\text{-}name}\ DepositsNY \bowtie Branch$$

# Relational Algebra 2

**Customer** (CustID, Name, street, city, zip)

**Deposit** (CustID, Acct-num, balance, Branch-name)

**Branch** (Branch-name, assets, Branch-city)

Find assets and name of all branches that have depositors (i.e., customers with a Deposit account) living in New York.

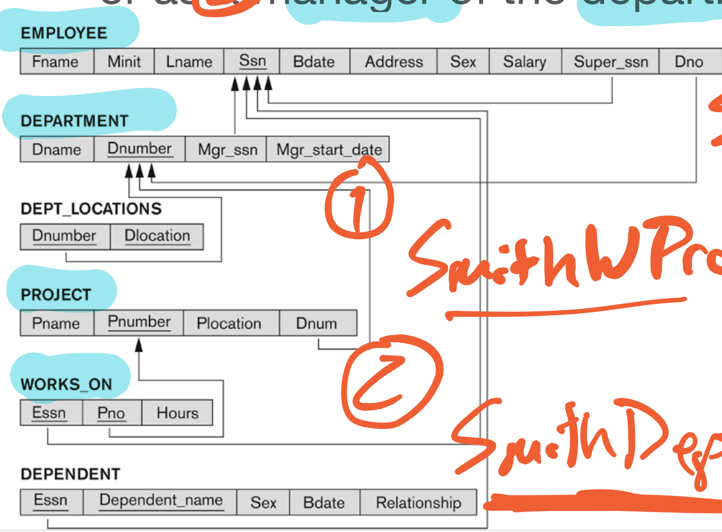
$$NYCustomer \leftarrow \sigma_{city=New\ York}\ Customer$$

$$NYDeposit \leftarrow NYCusts \bowtie Deposit$$

$$\Pi_{branch-name,\ assets}\ Branch \bowtie NYDeposit$$

# Relational Algebra 3

Find the list of all project numbers for projects that involve an employee whose last name is Smith, either as a worker (i.e., works on the project) or as a manager of the department that controls the project.




$Smiths \leftarrow \sigma_{lname=Smith}\, Employee$

$SmithWProj \leftarrow Works\_on \bowtie_{essn=ssn} Smiths$

$SmithDepts \leftarrow Smiths \bowtie_{mgr\_ssn=ssn} Dept$

$SmithMgrProj \leftarrow SmithDept \bowtie_{dnum=dnumber} Project$

$\pi_{pnum}\, SmithMgrProjs \cup \pi_{pnum}\, SmithWProjs$

# Relational Algebra 3

Find the list of all project numbers for projects that involve an employee whose last name is Smith, either as a worker (i.e., works on the project) or as a manager of the department that controls the project.

Find all the Smiths, then find Smith worker projects
Find all departments run by smiths, then find all projects run by those departments
Combine these together for final result

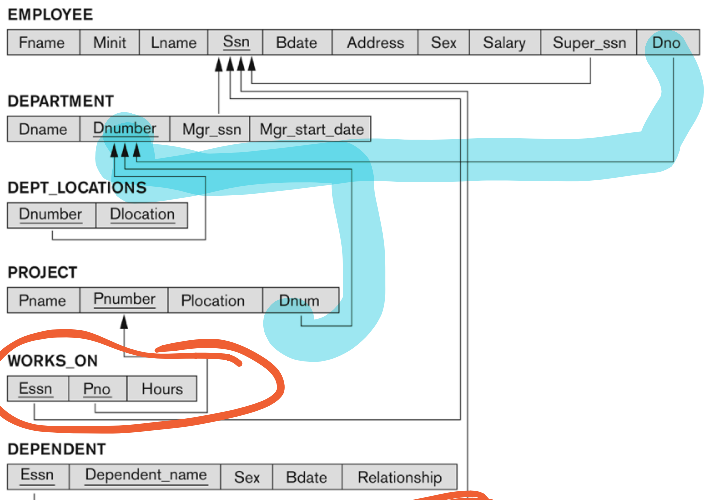
$$Smiths \leftarrow \sigma_{lname='Smith'}(Employee)$$
$$SmithWorkerProjs \leftarrow \pi_{pno}(WorksOn \bowtie_{essn=ssn} Smiths))$$
$$SmithDepts \leftarrow (Department \bowtie_{Mgr_s sn=Ssn} Smiths)$$
$$SmithMgrProjs \leftarrow \pi_{pno}(SmithDpets \bowtie_{Dnumber=Dnum} Project)$$
$$Answer \leftarrow SmithWorkerProjs \bigcup SmithMgrProjs$$

Any other questions on Relational Algebra?

# Next: SQL Queries

# SQL HW

7. Retrieve the list of employees, the projects they are working on, and their salary, ordered by department and, within each department, listed by decreasing salary, then alphabetically by last name, first name.




**Why is this wrong?**

```
select dname,lname,fname,pname,salary
from   employee
JOIN department on department.dnumber=employee.dno
JOIN project on project.Dnum = Employee.dno
order by dname,salary DESC, lname, fname;
```

# SQL HW

7. Retrieve the list of employees, the projects they are working on, and their salary, ordered by department and, within each department, listed by decreasing salary, then alphabetically by last name, first name.

Just because a project is in a department, doesn't mean that employee works on it! Need to join using the works_on table.

```
select dname,lname,fname,pname,salary
from   department
JOIN employee on department.dnumber=employee.dno
JOIN works_on on works_on.essn = employee.ssn
JOIN project on project.pnumber = works_on.pno
order by dname,salary DESC, lname, fname
```

Any other questions on SQL?

# Next: Normalization

# Normal Forms - more definitions

2NF: A schema is in 2NF if

- No nonprime attribute is partially dependent on the candidate key (i.e., depends on only part of a candidate key)

3NF: A schema is in 3NF if

- It is in 2NF and,
- no nonprime attribute is transitively dependent on the primary key (LHS must be a full key, unless RHS is a key)

BCNF: A schema is in BCNF if

- It is in 3NF and,
- If RHS is a key, LHS must be a super key

# Normalization - Finding Keys

Q5b) Consider the relation *R3 = (A, B, C, D)*, with the following functional dependencies:

- **AB -> C** and **C -> D**

What is the Candidate Key for this relation? What normal form does *R3* satisfy? You may assume that all tuples are unique and attributes are atomic.

AB -> C    C -> D

2NF ✓    Not 3NF

C -> D violates 3NF

# Normalization - Finding Keys

Q5b) Consider the relation *R3 = (A, B, C, D)*, with the following functional dependencies:

- **AB -> C** and **C -> D**

What is the Candidate Key for this relation? What normal form does *R3* satisfy? You may assume that all tuples are unique and attributes are atomic.

Candidate Key is AB since:
AB -> C and
AB -> C -> D
so, with AB we can determine all attributes

Normal form is 2NF since C->D violates 3NF

# Decomposition

Q6 Suppose we decompose Relation **R5** into two tables, **R51** and **R52**:

- **R51 = (V, W, Y, Z)**
- **R52 = (V, W, X)**

Will this be a loss-free decomposition, i.e., will we still be able to reconstruct all data by joining the two tables together? What normal form will *R51* and *R52* be in?

R5 = (**V**, **W**, X, **Y**, Z)

V -> X
WY -> X
VWY -> Z

# Decomposition

Q6 Suppose we decompose Relation **R5** into two tables, **R51** and **R52**:

- **R51 = (V, W, Y, Z)**
- **R52 = (V, W, X)**

Will this be a loss-free decomposition, i.e., will we still be able to reconstruct all data by joining the two tables together? What normal form will *R51* and *R52* be in?

R5 = (<u>V</u>, <u>W</u>, X, <u>Y</u>, Z)

V -> X
WY -> X
VWY -> Z

## Lossless Decomposition test: (from normalization lecture 2)

- **R1**, **R2** is a lossless join decomposition of **R** with respect to **F** **iff** at least one of the following dependencies is in **F+**

① **(R1 ∩ R2) → R1 – R2**

② **(R1 ∩ R2) → R2 – R1**

- R51 ∩ R52 = V W
- R51 – R52 = Y Z
- R52 – R51 = X

# Decomposition

Q6 Suppose we decompose Relation **R5** into two tables, **R51** and **R52**:

- **R51 = (V, W, Y, Z)**
- **R52 = (V, W, X)**

R5 = (V, W, X, Y, Z)

V -> X

WY -> X

VWY -> Z

## Lossless Decomposition test:

(from normalization lecture 2)

- **R1**, **R2** is a lossless join decomposition of **R** with respect to **F** **iff** at least one of the following dependencies is in **F+**
- **(R1 ∩ R2) → R1 – R2** VW -> YZ
- **(R1 ∩ R2) → R2 – R1** VW -> X

R51 ∩ R52 = VW

R51 - R52 = YZ

R52 - R51 = X

VW -> X is part of F+

# Decomposition

Q6 Suppose we decompose Relation **R5** into two tables, **R51** and **R52**:

- **R51 = (V, W, Y, Z)**
- **R52 = (V, W, X)**

VWY -> Z

V -> X

Will this be a loss-free decomposition, i.e., will we still be able to reconstruct all data by joining the two tables together? **What normal form will *R51* and *R52* be in?**

R5 = (**V**, **W**, X, **Y**, Z)

V -> X
WY -> X
VWY -> Z

R51 -> 3NF

R52 -> 1NF

# Decomposition

Q6 Suppose we decompose Relation **R5** into two tables, **R51** and **R52**:

- **R51 = (V, W, Y, Z)**
- **R52 = (V, W, X)**

Will this be a loss-free decomposition, i.e., will we still be able to reconstruct all data by joining the two tables together? **What normal form will *R51* and *R52* be in?**

R5 = (**V**, **W**, X, **Y**, Z)

V -> X
WY -> X
VWY -> Z

R51 is 3NF since only VWY->Z holds and VWY is the full candidate key

R52 is 1NF since V->X holds and V is a partial candidate key, so it cannot be 2NF

# Decomposition

Q6 Suppose we decompose Relation **R5** into two tables, **R51** and **R52**:

- **R51 = (V, W, Y, Z)**
- **R52 = (V, W, X)**

How can we decompose and ensure 3NF for all relations?

R5 = (**V**, **W**, X, **Y**, Z)

V -> X

WY -> X

VWY -> Z

V -> X

R53 = WYX + VWYZ

# Decomposition

Q6 Suppose we decompose Relation **R5** into two tables, **R51** and **R52**:

- **R51 = (V, W, Y, Z)**
- **R52 = (V, W, X)**

How can we decompose and ensure 3NF for all relations?

R5 = (**V**, **W**, X, **Y**, Z)

V -> X
WY -> X
VWY -> Z

R51 is already 3NF

To fix R52 we could use
R53 = (V, X)
This must be 3NF

R51 ∩ R53 = V
R51 - R53 = WYZ
R53 - R51 = X

V -> X is part of F+

# Clarification from lecture

In lecture we discussed the problem at right and I briefly second guessed myself on whether

VW->X is part of F+

the answer is that **yes it is**, because we are given that **V -> X**, and from the **Augmentation** rule we can get **VW -> WX**
Then from the **Decomposition** rule, we can get
**VW->X** and **VW -> W** are part of F+

R5 = (V, W, X, Y, Z)

F =
V -> X
WY -> X
VWY -> Z

R51 ∩ R52 = VW
R51 - R52 = YZ
R52 - R51 = X

VW -> X is part of F+

Any other questions on Normalization?

# Next: Shopping Cart

# Shopping Cart Tips

Carefully read spec

- Make a list of tasks and workflows to test

Implement the tables from our ER diagram

- You may not have created tables or used foreign keys before!

Plan mockups of pages you will need

- Start with simplest requirements!
- Don't worry about making it pretty until later

If your code won't run… fix it!

- Don't try to write a lot of code without testing