Discussion 3

Relational Algebra Cont., SQL DML, & Java Intro

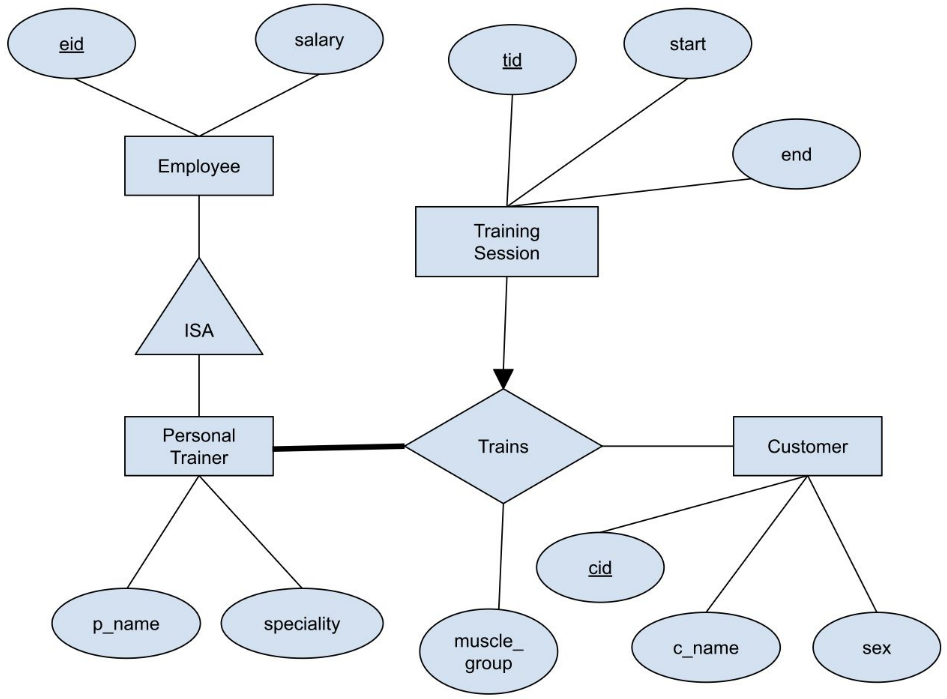
EECS 484

Logistics

- Project 1
  - Due Sep 24th, 11:45 PM ET
  - Groups of 2. Make sure to add each other as a group before submitting to the Gradescope and Autograder
- Homework 2
  - Due Oct 4th, 11:45 PM ET
  - Individual, No Groups!
- Project 2
  - will be released next Monday
  - Due Oct 22nd, 11:45 PM ET
  - Groups of 2

Ternary Relationship in HW1

The following ER diagram shows the relationships between a personal trainer and customers at a gym.



Ternary Relationship:

Training Session, Personal Trainer, and Customer

Question:

Both personal trainers Braden (eid=4) and Zach (eid=6) can train customer Lili (cid=12) in the same training session (tid=10).

True / False

Relational Algebra - Joins

Join

- Way to combine information from two tables with correlation
- Conditional join
  - RA: Relation1 $\bowtie_{condition}$ Relation2
    - Equivalent to $\sigma_{condition}$(Relation1 X Relation2)
    - condition can include multiple expressions
  - Equijoin is a conditional join with restrictions on condition
    - Only equalities between fields and $\wedge$ connectors

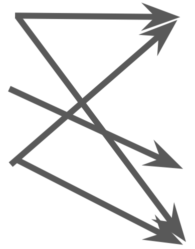
Join

- Way to combine information from two tables with correlation
- Natural join (without specifying condition)
  - Relation1 $\bowtie$ Relation2
  - Equijoin but automatic on all columns with the same name (must be same type)
  - Duplicate columns are dropped

Course ⋈ Student (Example of natural join)

Course

CID	Major
575	CS
548	Aerospace
484	CS
412	Math



Student

UserID	Name	Major	Drink
1	Alice	CS	Tea
2	Bob	Undeclared	Tea
3	Cathy	Aerospace	Tea
4	Dylan	CS	Coffee

CID	Major	UserID	Name	Drink
575	CS	1	Alice	Tea
484	CS	1	Alice	Tea
548	Aero	3	Cathy	Tea
575	CS	4	Dylan	Coffee
484	CS	4	Dylan	Coffee

Set Operators Summary for Your Reference

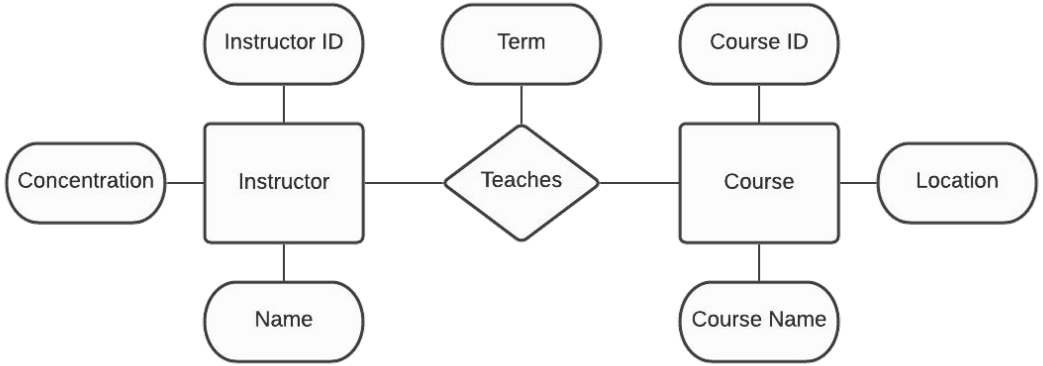
Union	$\text{Relation1} \cup \text{Relation2}$
Intersection	$\text{Relation1} \cap \text{Relation2}$
Set Difference	$\text{Relation1} - \text{Relation2}$
Cross Product	$\text{Relation1} \times \text{Relation2}$

- Relations must be “compatible” to perform union, intersection, or set difference (but **not** cross product)
- What is “compatibility”?:
  - Relations have the **same number of fields**
  - Corresponding **fields** are of the **same datatype**

$$\sigma_{\text{topic='RA'}}(\text{PracticeProblems})$$

Example Problem

- Three tables:
    - Courses, Instructors, Teaches



Course ID	Course Name	Location
EECS 575	Crypto	1690BBB
EECS 484	Databases	1670BBB
EECS 482	OS	1690BBB
EECS 388	Security	404BBB
AERO 548	Astrodynamics	FXB1012
EECS 999	Redacted	???

Instructor ID	Name	Concentration
1	Alice	Cryptography
2	Bob	Boring
3	SQL	Databases
4	Eve	Hacking :D
5	Buzz	Space
6	Mr. Meow	Meowing

Instructor ID	Course ID	Term
2	EECS 575	F20
3	EECS 484	F20
1	EECS 482	W19
2	AERO 548	W19
3	EECS 388	W19
4	EECS 388	W21

Q1

- Write the RA statement to select the names of all the instructors who teach in 1690 BBB and the courses that they teach
    - Don’t select courses that have no instructor or instructors that don’t teach anything

Course ID	Course Name	Location
EECS 575	Crypto	1690BBB
EECS 484	Databases	1670BBB
EECS 482	OS	1690BBB
EECS 388	Security	404BBB
AERO 548	Astrodynamics	FXB1012
EECS 999	Redacted	???

Instructor ID	Name	Concentration
1	Alice	Cryptography
2	Bob	Boring
3	SQL	Databases
4	Eve	Hacking :D
5	Buzz	Space
6	Mr. Meow	Meowing

Instructor ID	Course ID	Term
2	EECS 575	F20
3	EECS 484	F20
1	EECS 482	W19
2	AERO 548	W19
3	EECS 388	W19
4	EECS 388	W21

Q1

- Write the RA statement to select the names of all the instructors who teach in 1690 BBB and the courses that they teach
  - Don’t select courses that have no instructor or instructors that don’t teach anything
- $\pi_{\text{name, Course\_ID}}(\text{Instructor} \bowtie \text{Teaches} \bowtie \sigma_{\text{Location='1690 BBB'}}(\text{Course}))$ OR
- $\pi_{\text{name, Course\_ID}}(\sigma_{\text{Location='1690 BBB'}}(\text{Instructor} \bowtie \text{Teaches} \bowtie \text{Course}))$
- Why do I not need conditions on the joins?

Q1

- Write the RA statement to select the names of all the instructors who teach in 1690 BBB and the courses that they teach
  - Don’t select courses that have no instructor or instructors that don’t teach anything
- $\pi_{\text{name, Course\_ID}}(\text{Instructor} \bowtie \text{Teaches} \bowtie \sigma_{\text{Location='1690 BBB'}}(\text{Course}))$ OR
- $\pi_{\text{name, Course\_ID}}(\sigma_{\text{Location='1690 BBB'}}(\text{Instructor} \bowtie \text{Teaches} \bowtie \text{Course}))$
- Why do I not need conditions on the joins?
  - No condition implies natural join

Q2

- Here are two RA statements that finds all instructor ids (iid) who taught in terms F20 and W19. What is incorrect about each one?

1. Teaches / $\pi_{term}(\sigma_{term='F20' \vee term='W19'}(\text{Teaches}))$
   a.

1. $\pi_{iid}(\ \rho(T1, \text{Teaches}) \bowtie_{T1.iid=T2.iid \wedge T1.term='F20' \wedge T2.term='W19'} \rho(T2, \text{Teaches})$
   $)$
   b.

iid	cid	term
2	EECS 575	F20
3	EECS 484	F20
1	EECS 482	W19
2	AERO 548	W19
3	EECS 484	W19
4	EECS 388	W21

Q2

- Here are two RA statements that finds all instructor ids (iid) who taught in terms F20 and W19. What is incorrect about each one?

1. Teaches / $\pi_{term}(\sigma_{term='F20' \vee term='W19'}(\text{Teaches}))$
   a. $\pi_{iid,\ term}(\text{Teaches}) / \pi_{term}(\sigma_{term='F20' \vee term='W19'}(\text{Teaches}))$

1. $\pi_{iid}(\ \rho(T1, \text{Teaches}) \bowtie_{T1.iid=T2.iid \wedge T1.term='F20' \wedge T2.term='W19'} \rho(T2, \text{Teaches})$
   $)$
   b.

iid	cid	term
2	EECS 575	F20
3	EECS 484	F20
1	EECS 482	W19
2	AERO 548	W19
3	EECS 484	W19
4	EECS 388	W21

Q2

- Here are two RA statements that finds all instructor ids (iid) who taught in terms F20 and W19. What is incorrect about each one?

1. Teaches / $\pi_{term}(\sigma_{term='F20' \vee term='W19'}(\text{Teaches}))$
    a. $\pi_{iid,\ term}(\text{Teaches}) / \pi_{term}(\sigma_{term='F20' \vee term='W19'}(\text{Teaches}))$

1. $\pi_{iid}(\ \rho(T1, \text{Teaches}) \bowtie_{T1.iid=T2.iid \wedge T1.term='F20' \wedge T2.term='W19'} \rho(T2, \text{Teaches})\ )$
    b. $\pi_{T1.iid}(\ \rho(T1, \text{Teaches}) \bowtie_{T1.iid=T2.iid \wedge T1.term='F20' \wedge T2.term='W19'} \rho(T2, \text{Teaches})\ )$

iid	cid	term
2	EECS 575	F20
3	EECS 484	F20
1	EECS 482	W19
2	AERO 548	W19
3	EECS 484	W19
4	EECS 388	W21

Q2

- Another solution using intersection if you want to take a look :)

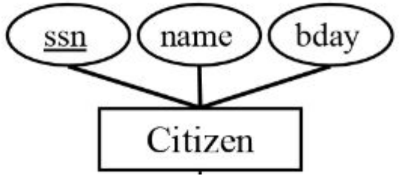
$(\pi_{iid}(\sigma_{term='F20'}(Teaches))) \cap (\pi_{iid}(\sigma_{term='W19'}(Teaches)))$

iid	**cid**	**term**
2	EECS 575	F20
3	EECS 484	F20
1	EECS 482	W19
2	AERO 548	W19
3	EECS 484	W19
4	EECS 388	W21

SQL – Data Manipulation Language

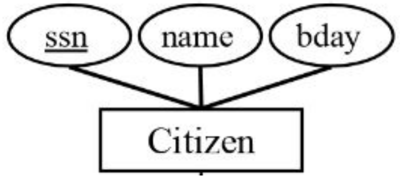
Select

- SELECT column_1, column_2, …
  FROM Table_Name
  WHERE condition_1 AND condition_2 …;
  - Selects data from the table. Can choose which columns you want
  - Where clause is conditional
    - Only choose data that satisfies entire clause (can use ands and ors)
- Example:
  - SELECT name, bday FROM Citizen
    WHERE (name = ‘John’ OR name = ‘Jane’)
    AND bday = TO_DATE('1998-DEC-25','YYYY-MON-DD');
- Can SELECT DISTINCT specifically
  - Removes all duplicates



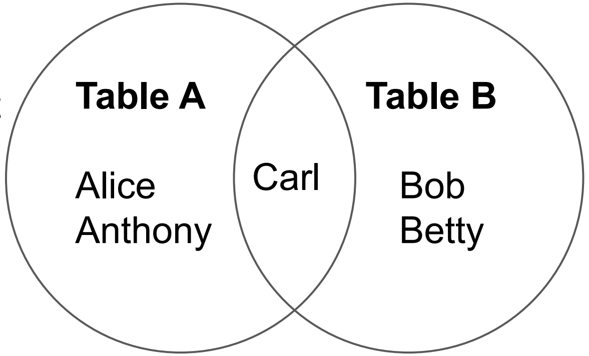
Insert

- INSERT INTO Table_Name (column_1, column_2, …)
  VALUES (value_for_column_1, value_for_column_2, …);
  - Inserts data mapping values to the columns
  - Take care to ensure all necessary columns are populated and all data is valid
- Example:
  - INSERT INTO Citizen (ssn, name)
    VALUES (123456789, ‘Bob’);
- Can also INSERT from SELECT statement
  - INSERT INTO Citizen (ssn, name, bday)
    SELECT ssn, name, bday
    FROM public_schema.Public_Citizens
    WHERE (name = ‘John’ OR name = ‘Jane’);



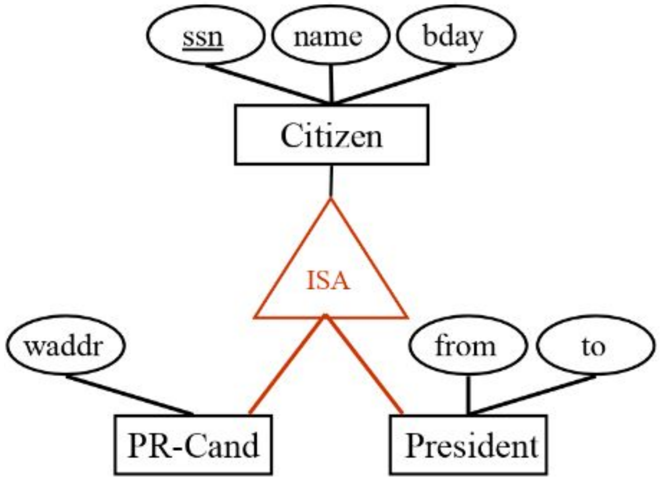
Union, Minus, Intersect

- Set operations
  - Union adds two sets and finds everything that is in either or
  - Minus subtracts everything in the second set from the first set
  - Intersection takes everything that is in both sets
- SELECT name FROM Table_A
  - Alice, Anthony, Carl
- SELECT name FROM Table_B
  - Bob, Betty, Carl
- SELECT name FROM Table_A UNION SELECT name FROM Table_B
  - Alice, Anthony, Carl, Bob, Betty
- SELECT name FROM Table_A MINUS SELECT name FROM Table_B
  - Alice, Anthony
- SELECT name FROM Table_A INTERSECT SELECT name FROM Table_B
  - Carl



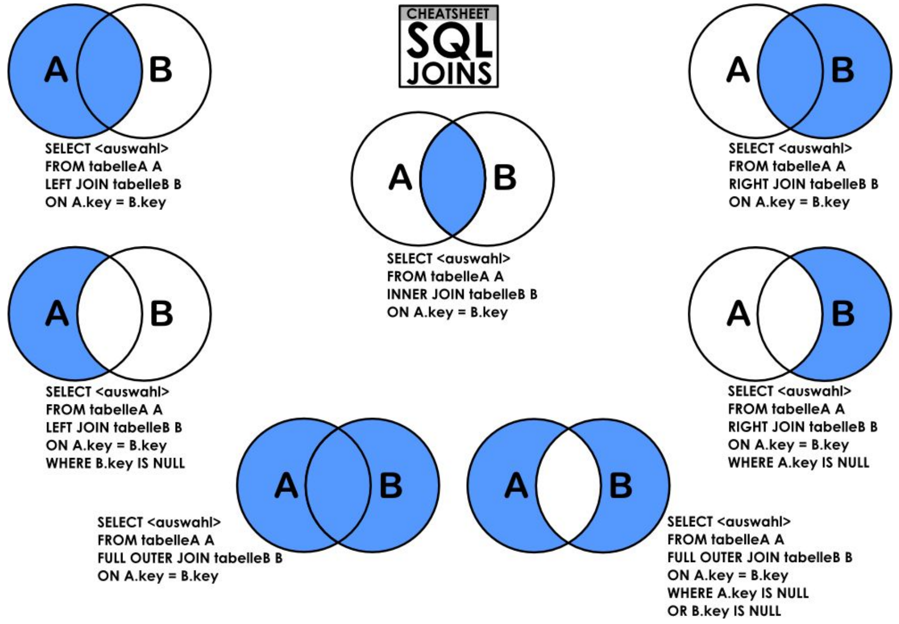
Views

- Views provide a lookup on a pre-established query
    - Define for the database what data you would like to see and stores query
    - Associated lookup run each time the view is accessed
    - Part 4 of the project needs views
- CREATE VIEW Presidents_View AS
  SELECT name, from, to
  FROM President;



Joins

- Joins allow us to merge data across tables into one large table
  - Super powerful, but can be complex
  - Can have joins across multiple tables
  - Useful when you need to correlate data
  - Different types of joins
  - Necessary for Part 3 of project 1



```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```

```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```

```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL
```

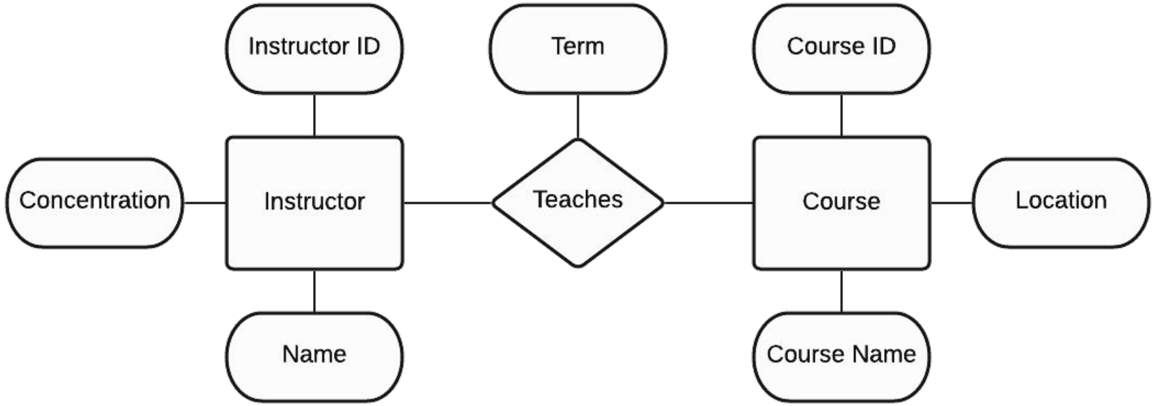
```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
```

```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
```

```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

`INSERT INTO` PracticeProblems

Example Problem



Instructor

InstructorID	Name	Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

Teaches

InstructorID	CourseID	Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

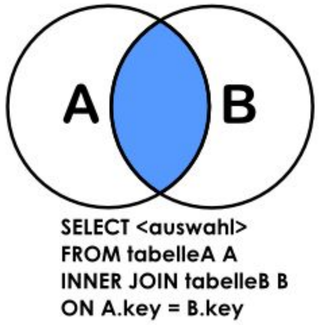
Course

CourseID	CourseName	Location
EECS 575	Crypto	1690BBB
EECS 484	Databases	1670BBB
EECS 482	OS	1690BBB
EECS 999	Redacted	somewhere

Inner Join

- Joins entries when the join condition is satisfied
  - Could have WHERE clause too

```
SELECT I.Name, T.CourseID, T.Term
FROM Instructor I
INNER JOIN Teaches T
ON I.InstructorID = T.InstructorID;
```

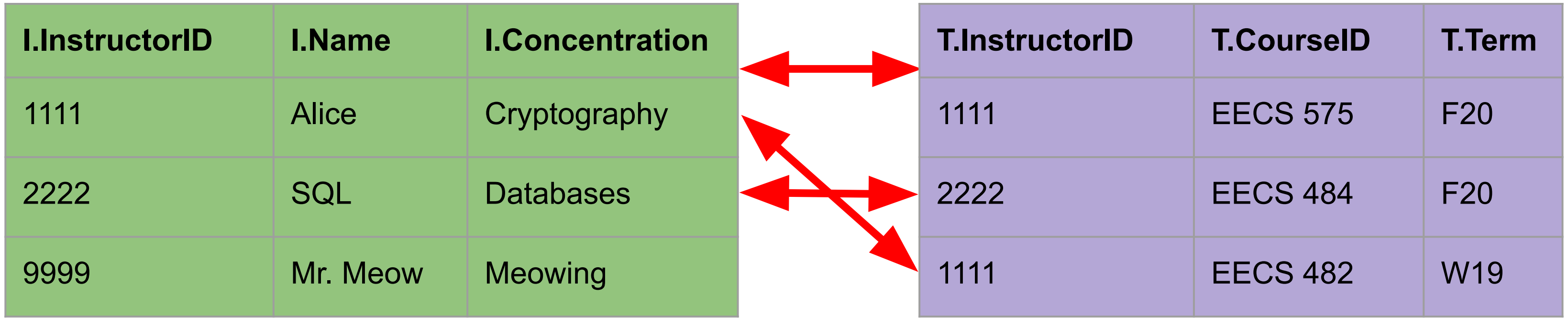


```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```

I.InstructorID	I.Name	I.Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

Inner Join



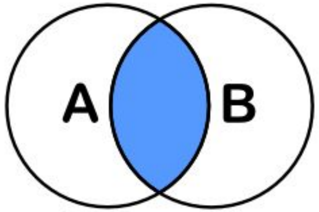
I.InstructorID	I.Name	I.Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

I.InstructorID	I.Name	I.Concentration	T.InstructorID	T.CourseID	T.Term
1111	Alice	Cryptography	1111	EECS 575	F20
2222	SQL	Databases	2222	EECS 484	F20
1111	Alice	Cryptography	1111	EECS 482	W19

Inner Join

- Where clause syntax
  - SELECT I.Name, T.CourseID, T.Term
    FROM Instructor I, Teaches T
    WHERE I.InstructorID = T.InstructorID;



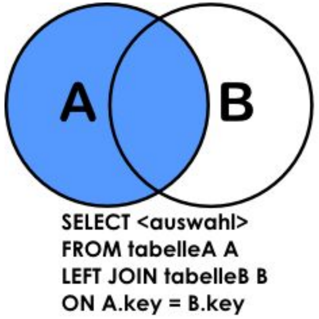
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key

I.Name	**T.CourseID**	**T.Term**
Alice	EECS 575	F20
SQL	EECS 484	F20
Alice	EECS 482	W19

Left Join

- Entries in left table with no entry in right table get null for right table columns
  - Rest is same as inner join

```
SELECT I.Name, T.CourseID, T.Term
FROM Instructor I
LEFT JOIN Teaches T
ON I.InstructorID = T.InstructorID;
```

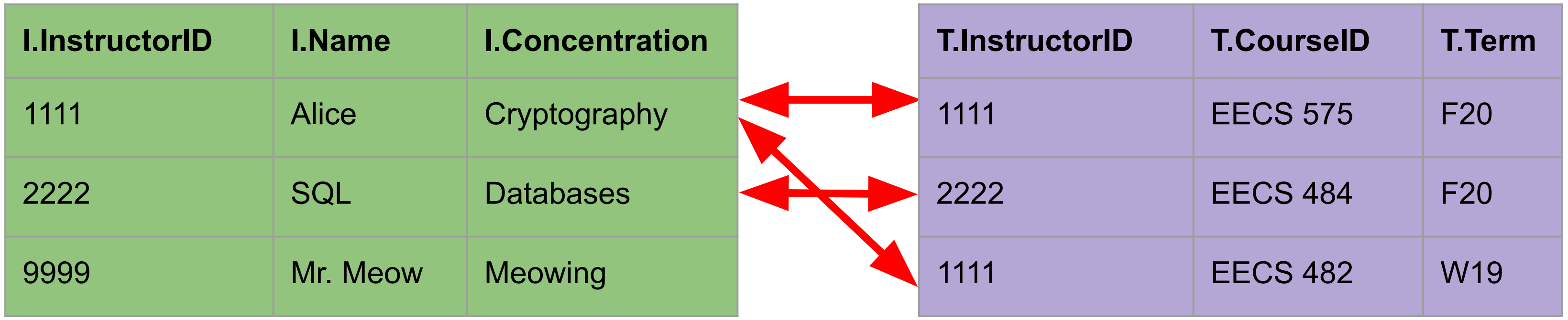


```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```

I.InstructorID	I.Name	I.Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

Left Join



I.InstructorID	I.Name	I.Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

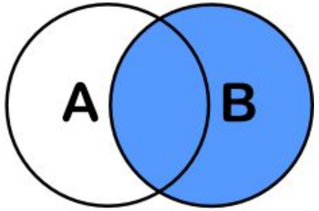
T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

I.InstructorID	I.Name	I.Concentration	T.InstructorID	T.CourseID	T.Term
1111	Alice	Cryptography	1111	EECS 575	F20
2222	SQL	Databases	2222	EECS 484	F20
1111	Alice	Cryptography	1111	EECS 482	W19
9999	Mr. Meow	Meowing	NULL	NULL	NULL

Right Join

- Entries in right table with no entry in left table get null for left table columns
  - Rest is same as inner join

```
SELECT I.Name, T.CourseID, T.Term
FROM Teaches T
RIGHT JOIN Instructor I
ON I.InstructorID = T.InstructorID;
```

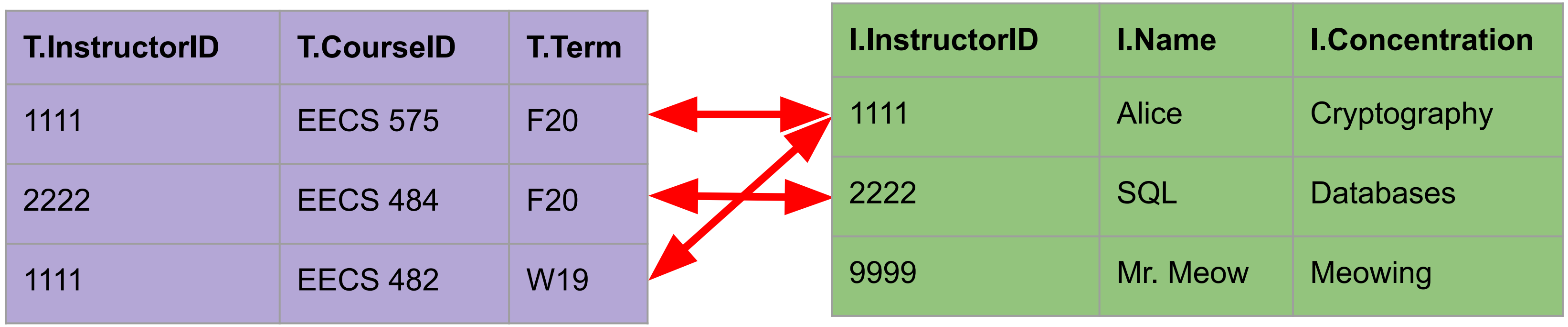


```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```

T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

I.InstructorID	I.Name	I.Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

Right Join



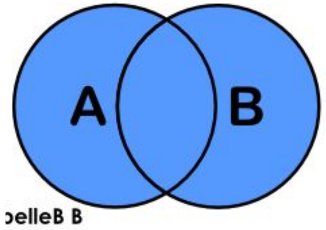
T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

I.InstructorID	I.Name	I.Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

T.InstructorID	T.CourseID	T.Term	I.InstructorID	I.Name	I.Concentration
1111	EECS 575	F20	1111	Alice	Cryptography
2222	EECS 484	F20	2222	SQL	Databases
1111	EECS 482	W19	1111	Alice	Cryptography
NULL	NULL	NULL	9999	Mr. Meow	Meowing

Full Outer Join

- All entries combined with non-corresponding ones null
  - SELECT I.Name, C.Course_Name, C.Course_ID, C.Location, T.Term
    FROM Instructor I
    OUTER JOIN Teaches T
    ON I.Instructor_ID = T.Instructor_ID
    OUTER JOIN Course C
    ON T.Course_ID = C.Course_ID;



InstructorID	Name	Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

InstructorID	CourseID	Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

CourseID	CourseName	Location
EECS 575	Crypto	1690BBB
EECS 484	Databases	1670BBB
EECS 482	OS	1690BBB
EECS 999	Redacted	somewhere

Full Outer Join

```
SELECT I.Name, C.Course_Name, C.Course_ID, C.Location, T.Term
FROM Instructor I
OUTER JOIN Teaches T
ON I.Instructor_ID = T.Instructor_ID
OUTER JOIN Course C
ON T.Course_ID = C.Course_ID;
```

Name	**Course Name**	**Course ID**	**Course Location**	**Term**
Alice	Crypto	EECS 575	1690BBB	F20
SQL	Databases	EECS 484	1670BBB	F20
Alice	OS	EECS 482	1690BBB	W19
NULL	Redacted	EECS 999	somewhere	NULL
Mr.Meow	NULL	NULL	NULL	NULL

Multiple Inner Joins Example

- Syntax 1

```
SELECT I.Name, C.CourseID, C.Location
FROM Instructor I
INNER JOIN Teaches T ON I.InstructorID = T.InstructorID
INNER JOIN Course C ON T.CourseID = C.CourseID
WHERE T.Term = ‘F20’;
```

- Syntax 2

```
SELECT I.Name, C.CourseID, C.Location
FROM Instructor I, Teaches T, C.CourseID
WHERE I.InstructorID = T.InstructorID
        AND T.CourseID = C.CourseID
        AND T.Term = ‘F20’;
```

InstructorID	Name	Concentration
1111	Alice	Cryptography
2222	SQL	Databases
9999	Mr. Meow	Meowing

InstructorID	CourseID	Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

CourseID	CourseName	Location
EECS 575	Crypto	1690BBB
EECS 484	Databases	1670BBB
EECS 482	OS	1690BBB
EECS 999	Redacted	somewhere

Multiple Inner Joins Example

Instructor I INNER JOIN Teaches T ON I.InstructorID = T.InstructorID

INNER JOIN Course C ON T.CourseID = C.CourseID

I.InstructorID	I.Name	I.Concentration	T.InstructorID	T.CourseID	T.Term
1111	Alice	Cryptography	1111	EECS 575	F20
2222	SQL	Databases	2222	EECS 484	F20
1111	Alice	Cryptography	1111	EECS 482	W19

C.CourseID	C.CourseName	C.Location
EECS 575	Crypto	1690BBB
EECS 484	Databases	1670BBB
EECS 482	OS	1690BBB
EECS 999	Redacted	???

SELECT I.Name, C.CourseID, C.Location

WHERE T.Term = ‘F20’

I.InstructorID	I.Name	I.Concentration	T.InstructorID	T.CourseID	T.Term	C.CourseID	C.CourseName	C.Location
1111	Alice	Cryptography	1111	EECS 575	F20	EECS 575	Crypto	1690BBB
2222	SQL	Databases	2222	EECS 484	F20	EECS 484	Databases	1670BBB
1111	Alice	Cryptography	1111	EECS 482	W19	EECS 482	OS	1690BBB

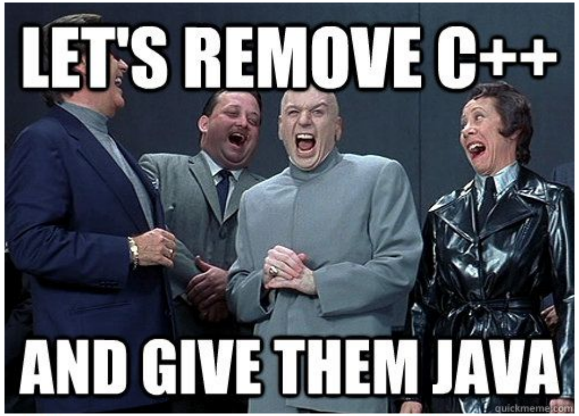
Java Tutorial

It’s not C++ ?!?!?!

Java Tutorial

- Java for P2!
- Java made by Oracle
    - We’ve been using Oracle DBMS so something tells us they’re a good match ;)

- Some differences from C++
    - No pointers!!!
        - **No** dynamic memory leaks!!!!
        - Automatic garbage collection
    - Some of the utility classes are weird
        - ArrayList, String
        - System.out.println instead of cout



When your teacher is talking about Java and you remember Minecraft was made with Java
You know, I'm something of a scientist myself.

Java Tutorial

- Java is really good at being used by a lot of different devices
    - Better than C++ for easy GUI development
        - Weak in comparison to C# but that only has support for Windows
- Object oriented to the extreme
    - Everything is a subclass of Object
- Java was designed to be used for the web
    - Uses packages to organize files
    - Historically packages are the domain name backwards
        - com.supersecretpage.eecs484iscool
- Instead of building an exe, java builds jars
    - You can open them up with 7Zip and look at all the files!
    - If you want an exe you have to use a tool like exe4j

Status: Installing Java

3 Billion Devices "Run" Java

versioning issues, slowness, unplanned downtime, crashes, support tickets, abandonware, unclear code, insecure web apps, unexplained outtages, interfacing problems, bugs

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;
ArrayList<Integer> numbers = new ArrayList<Integer>();	vector<int> numbers();

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;
ArrayList<Integer> numbers = new ArrayList<Integer>();	vector<int> numbers();
numbers.get(i);	numbers[i];

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;
ArrayList<Integer> numbers = new ArrayList<Integer>();	vector<int> numbers();
numbers.get(i);	numbers[i];
int [] numbersArray = new int[10];	int numbersArray[10];

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;
ArrayList<Integer> numbers = new ArrayList<Integer>();	vector<int> numbers();
numbers.get(i);	numbers[i];
int [] numbersArray = new int[10];	int numbersArray[10];
numbersArray[i];	numbersArray[i];

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;
ArrayList<Integer> numbers = new ArrayList<Integer>();	vector<int> numbers();
numbers.get(i);	numbers[i];
int [] numbersArray = new int[10];	int numbersArray[10];
numbersArray[i];	numbersArray[i];
String message = “Hello World”;	string message(“Hello World”);

Java Tutorial - Syntax Differences

Java	**C++**
System.out.println(“Hello World!”);	cout << “Hello World\n”;
ArrayList<Integer> numbers = new ArrayList<Integer>();	vector<int> numbers();
numbers.get(i);	numbers[i];
int [] numbersArray = new int[10];	int numbersArray[10];
numbersArray[i];	numbersArray[i];
String message = “Hello World”;	string message(“Hello World”);
boolean e = message.equals(“Hello World”);	bool e = message == “Hello World”;

Java Tutorial

- Objects act as if they are passed by reference*
    - Change an array that was passed in a function as a parameter and the original changes
- Primitives cannot be put into collections
    - ArrayList, etc.
    - Use a wrapper class instead
        - int -> Integer, double -> Double, boolean -> Boolean
    - Arrays work like expected however
- Most classes have ***capital letters***
    - String as opposed to string
- ***New*** keyword is ***everywhere***
    - Heap objects that are dynamically allocated but managed by Java so it's okay
- More info: https://docs.oracle.com/javase/7/docs/api/

*They're not actually passed by reference, but they exhibit a similar behavior

Get started with P1!

We’re here if you need any help!!

- Office Hours: Schedule is here, both virtual and in person offered
- Piazza
- Next week’s discussion!!!