Discussion 3

SQL

EECS 484

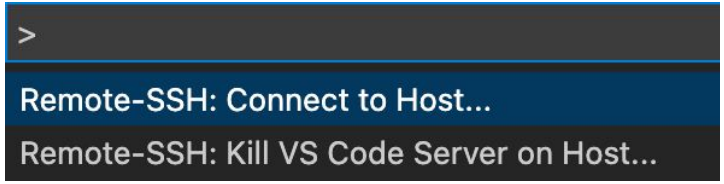
Logistics

- Homework 1 due today (Thursday Sept 15), 11:55 PM Eastern
- Project 1 due Sept. 22, 11:55 PM Eastern
- Midterm time: Oct. 21st 7:30 - 9:30 PM Eastern
    - Send an email to eecs484staff@umich.edu if you have a time conflict

SSH in VSCode troubleshooting

Several things to try:

1. Kill VS Code Server on Host
2. Modify the config file as in the following




```
# SSH multiplexing
Host *
    ControlMaster auto
    ControlPath ~/.ssh/master-%r@%h:%p

# CAEN
Host caen
    Hostname login.engin.umich.edu
    User <uniqname>
    ControlPersist 1h
```

SSH connection alternatives

1. Command line commands

   ssh <uniquename>@login.engin.umich.edu

2. Caen VNC client

3. Contact Caen help desk if you still need help

Enforcing Referential Integrity

Disallow deletion (default)

- **Disallow deletion** (default): If a tuple you are trying to delete is referred in another table, the record won't be deleted and instead will return an error message.
  - This is default behavior in Oracle, but ON DELETE NO ACTION in MySQL

```
CREATE TABLE Enroll (
        sno INT,
        cno INT,
        jdate date,
        PRIMARY KEY(sno,cno),
        FOREIGN KEY(sno) REFERENCES Student(sno)
        FOREIGN KEY(cno) REFERENCES Course(cno)
    );
```

delete

insert into students (1)

insert into enroll

(1, 2, ...)

delete from students

Delete all

only used for delete records

- **Delete all**: If a user deletes a row in the parent table, then the affected row is deleted in the child table.

  Example:

```
CREATE TABLE Enroll (
    sno INT,
    cno INT,
    jdate date,
    PRIMARY KEY(sno,cno),
    FOREIGN KEY(sno) REFERENCES Student(sno)
    ON DELETE CASCADE
    FOREIGN KEY(cno) REFERENCES Course(cno)
    ON DELETE CASCADE
);
```

delete student record

delete in Enroll table

delete students record
⇒ delete all records
there's nothing to this

Set to null

- **Set to null**: If a user deletes a row in the parent table, then the affected field is now set to null.
  - Example:

```
CREATE TABLE Enroll (
    sno INT,
    cno INT,
    jdate date,
    PRIMARY KEY(sno,cno),
    FOREIGN KEY(sno) REFERENCES Student(sno)
    ON DELETE SET NULL
    FOREIGN KEY(cno) REFERENCES Course(cno)
    ON DELETE SET NULL
);
```

DDL Practice Problems

Practice Problem

1. Take some time to write the SQL commands to create a table with the following schema:

- Table Name: Teas
- Columns: (column_name - type)
  i. Tea_Name - VARCHAR2(100)
  ii. Brew_Time - NUMBER
  iii. Brand - VARCHAR2(100)
- Constraints:
  i. Teas are stored by their primary key, the name of the tea
  ii. Each tea must have a brand

Practice Problem

1. Take some time to write the SQL commands to create a table with the following schema:

- Answer:

```
CREATE TABLE Teas (
        Tea_Name VARCHAR2(100) PRIMARY KEY,
        Brew_Time NUMBER,
        Brand VARCHAR2(100) NOT NULL
);
```

Practice Problem

2. Now that we have some tea, let’s make a menu table. Write the SQL statements to create the menu table:

- Table Name: Menus
- Columns: (column_name - type)
  i. Tea_Name - VARCHAR2(100)
  ii. Menu_Name - VARCHAR2(100)
  iii. Cost - NUMBER
- Constraints:
  i. Menu items are stored by Menu_Name as their primary key
  ii. Each menu item must have a Tea_Name which corresponds to an item in the Teas table
  iii. Each menu item must have a Cost

Practice Problem

2. Now that we have some tea, let’s make a menu table. Write the SQL statements to create the menu table:

- Answer:

```
CREATE TABLE Menus (
    Tea_Name VARCHAR2(100) NOT NULL,
    FOREIGN KEY (Tea_Name) REFERENCES Teas (Tea_Name),
    Menu_Name VARCHAR2(100) PRIMARY KEY,
    Cost NUMBER NOT NULL
);
```

Practice Problem

3. Finally, let’s drop all of our tables. Write the SQL Commands to drop them

Practice Problem

3. Finally, let's drop all of our tables. Write the SQL Commands to drop them

- DROP TABLE Menus;
  DROP TABLE Teas; refer X can't drop

  or

- DROP TABLE Teas CASCADE CONSTRAINTS;
  DROP TABLE Menus CASCADE CONSTRAINTS;

DML

data manipulation language

Select

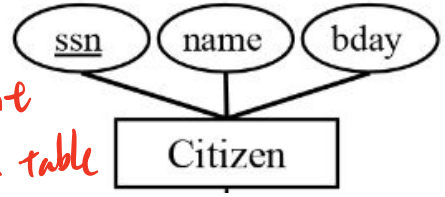
distinct → applies to all columns

John 1998/11/25

John 1998/11/25

⋮

- 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’) → can be omitted if you want all from table
    AND bday = TO_DATE('1998-DEC-25','YYYY-MON-DD');

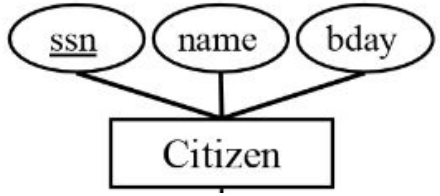


- Can SELECT DISTINCT specifically
  - Removes all duplicates

select col1, distinct col2 X

Insert

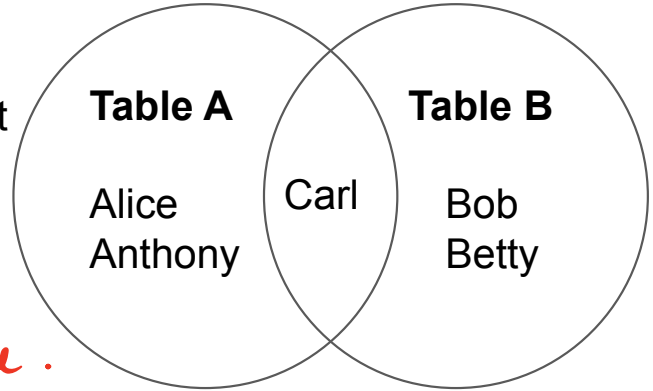
- INSERT INTO Table_Name (column_1, column_2, …)
  VALUES (value_for_column_1, value_for_column_2, …);
  - Inserts data mapping values to the columns
  - Take care to ensure all necessary columns are populated and all data is valid
- Example:
  - INSERT INTO Citizen (ssn, name)
    VALUES (123456789, ‘Bob’); ⇒ put NULL at bday.



- 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

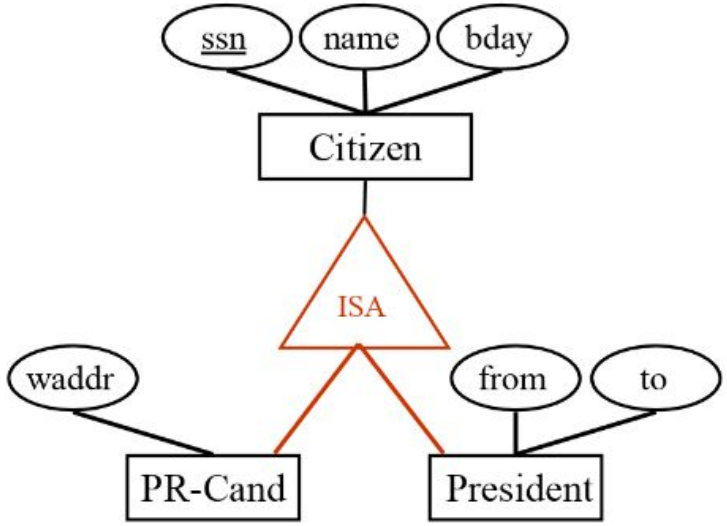
- Set operations
  - Union adds two sets and finds everything that is in either or
  - Minus subtracts everything in the second set from the first set
  - Intersection takes everything that is in both sets
- SELECT name FROM Table_A
  - Alice, Anthony, Carl
- SELECT name FROM Table_B
  - Bob, Betty,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



Compatible.

Views

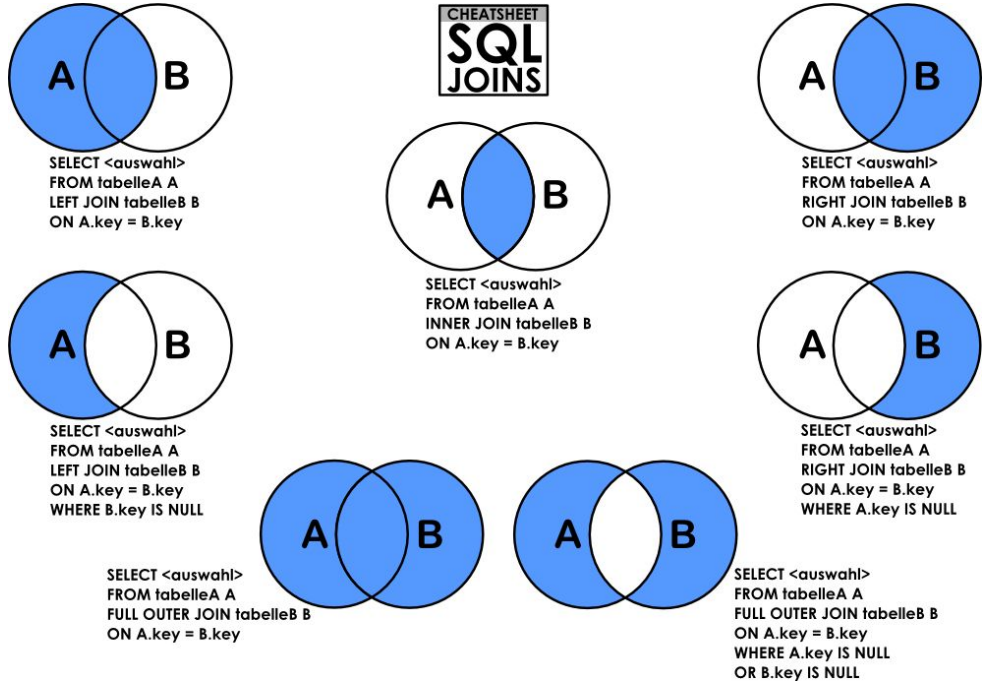
- Views provide a lookup on a pre-established query
  - Define for the database what data you would like to see and stores query
  - Associated lookup run each time the view is accessed
  - Part 4 of the project needs views
- CREATE VIEW Presidents_View AS
  SELECT name, from, to
  FROM President;



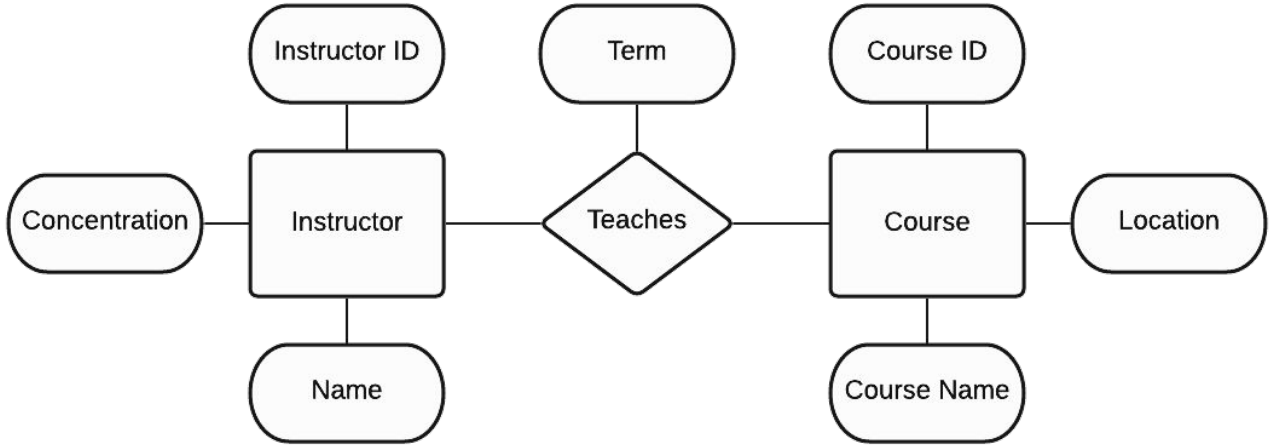
① select * from pre-view.
change
② ...
query view different

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



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

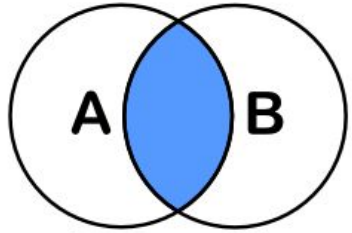
Course

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

Inner Join

⇒ requires attribute exist in both table

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



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

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

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

Inner Join

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

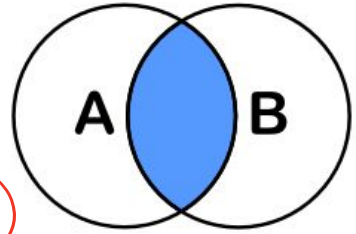
T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

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

I JOIN T ON xxx

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



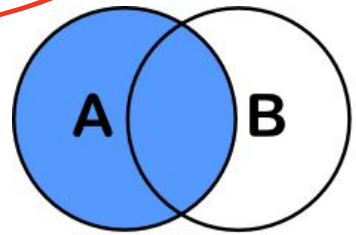
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key

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

left

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

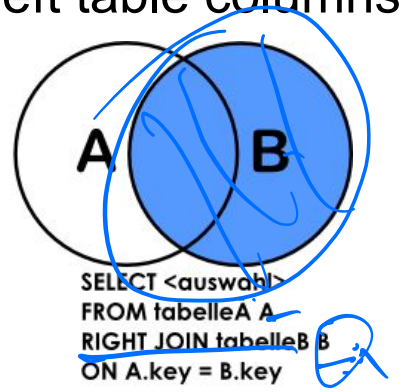
T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

null

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;
```

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

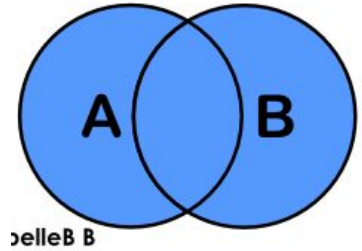
T.InstructorID	T.CourseID	T.Term
1111	EECS 575	F20
2222	EECS 484	F20
1111	EECS 482	W19

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

Get started with P1 Part 2, 3, 4!