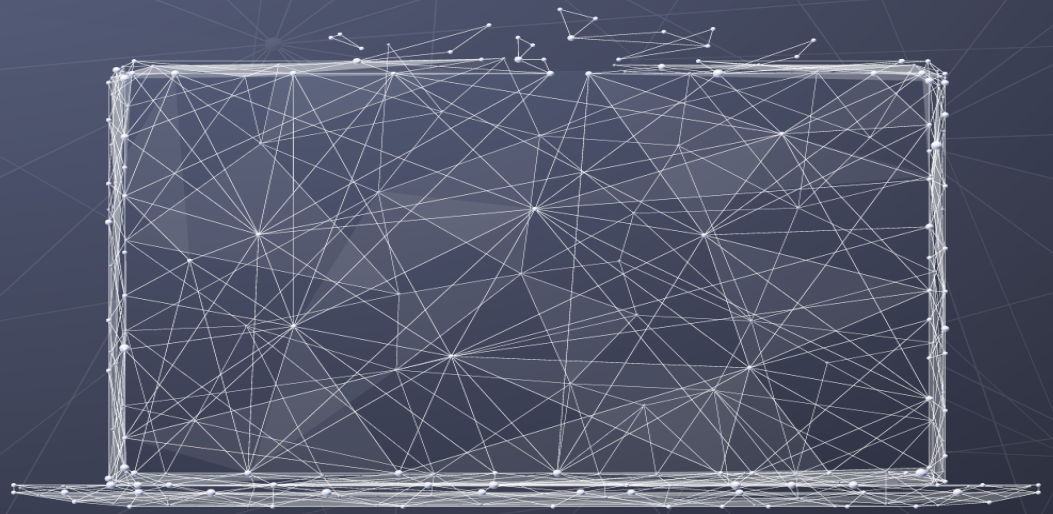
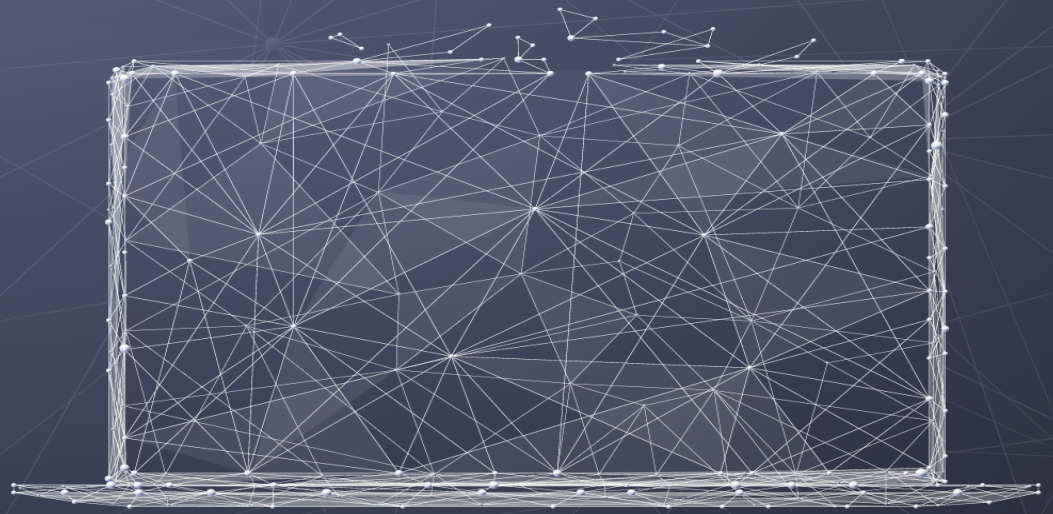
Data Science
"Data Engineering II"
Data Engineering II

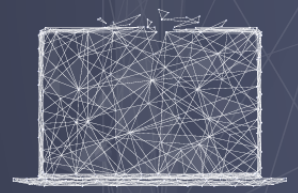


PURDUE UNIVERSITY | College of Science

Data Science
"Data Engineering II"
Mapping ER to Schema



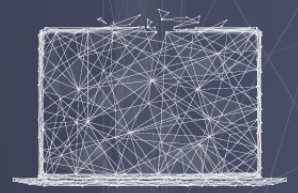
PURDUE UNIVERSITY | College of Science



Mapping from ER to Relational Model

Outcome

- In this module we will learn how to translate an ER diagram to a corresponding relational database schema
  - Mapping entities and relationships
  - Alternatives and their implications
  - How to enforce (some) constraints
  - Using SQL to create, alter or delete a table



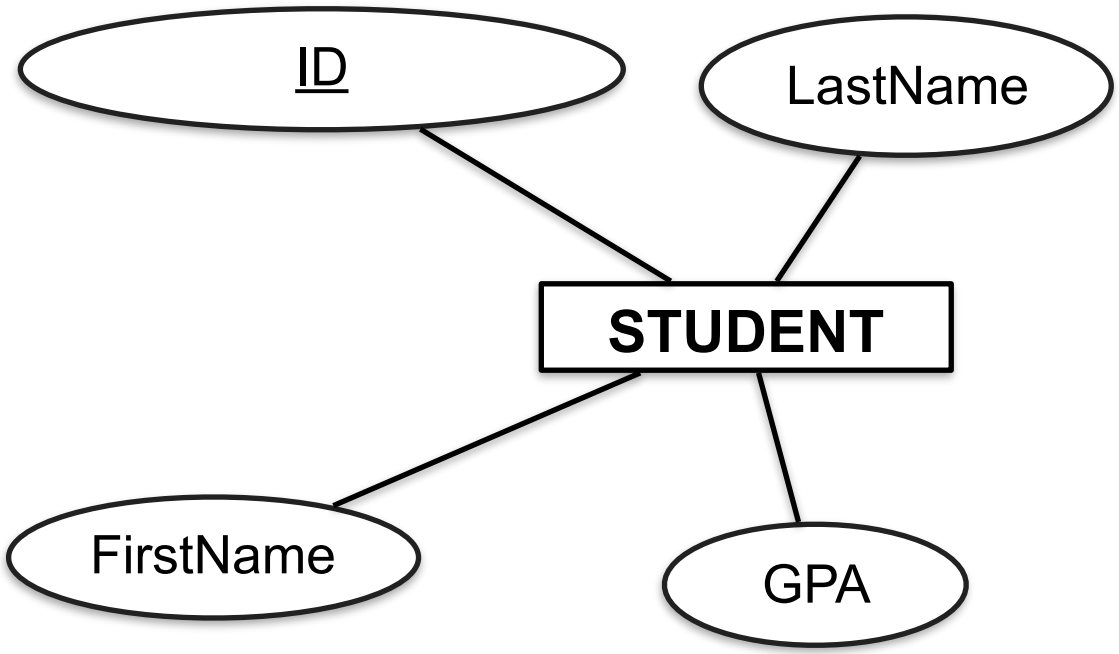
Mapping Strong Entities

Strong Entities

- For each strong entity, we typically create a single relation
  - The attributes include every attribute of the entity
  - We specify the domain of each attribute
  - Multivalued attributes require a separate table
  - We identify the primary key attributes
  - Candidate keys are specified too

Mapping Entities to Relations

The STUDENT Entity

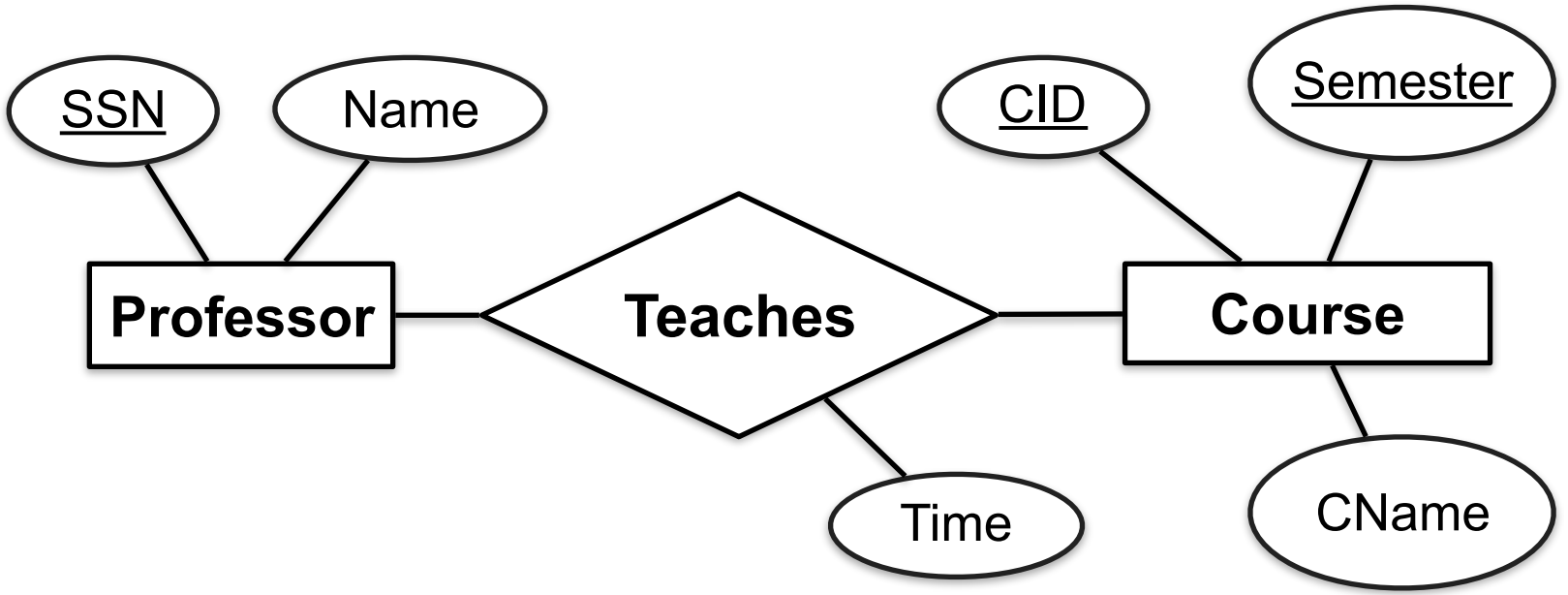


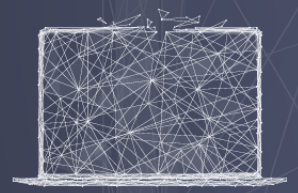
The STUDENT Relation

```
CREATE TABLE STUDENT (
    ID INTEGER,
    LastName VARCHAR(30),
    FirstName VARCHAR(30),
    GPA REAL,
    PRIMARY KEY (ID)
);
```

Mapping Relationships

Example Many-to-Many Relationship



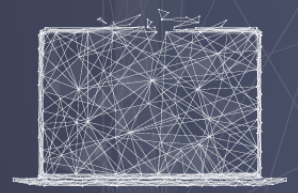


Mapping Relationships

The PROFESSOR and COURSE Relations

```
CREATE TABLE PROFESSOR (
    SSN INTEGER,
    NAME VARCHAR(30),
    PRIMARY KEY (SSN)
);
```

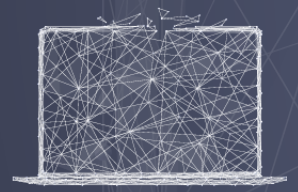
```
CREATE TABLE COURSE (
    CID INTEGER,
    CNAME VARCHAR(30),
    SEMESTER VARCHAR(30),
    PRIMARY KEY (CID, SEMESTER)
);
```



Mapping Relationships

Mapping Many to Many Relationships

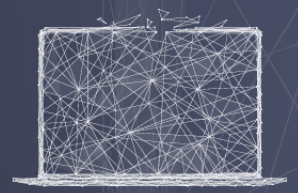
- Map the relationship to a new relation Teaches
  - Include the primary keys of each of the participating entities as attributes of Teaches
  - The union of these attributes will be the primary key for Teaches
  - Include the attributes of the relationship as attributes of relation Teaches



Mapping Relationships

The TEACHES Relation

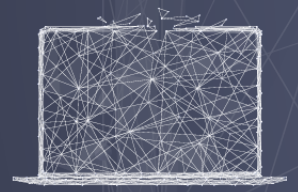
```
CREATE TABLE TEACHES (
    SSN INTEGER,
    CID INTEGER,
    SEMESTER VARCHAR(30),
    TIME DATETIME,
    PRIMARY KEY (SSN, CID, SEMESTER)
);
```



Mapping Relationships

Referential Integrity

- The previous solution has a problem:
  - It is possible to add invalid PROFESSORs and COURSEs in the TEACHES relation!
  - How do we prevent this? Only Professors in the PROFESSOR table should be allowed in this relationship
  - Similarly only courses in the COURSES table
- This requirement is called *Referential Integrity*
- We enforce this explicitly by declaring *Foreign Keys*



Sample Data Instance

PROFESSOR

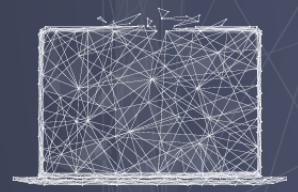
SSN	Name
2343	Sunil
45343	Aref
23432	Clifton

COURSE

CID	CName	Semester
CS54100	Databases	Fall 2019
CS54200	Dist. DB	Spr 2020

TEACHES

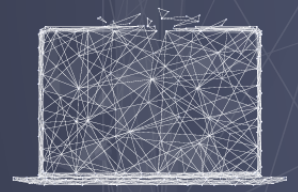
SSN	CID	Semester	Time
2343	CS54100	Fall 2019	9:00
45343	CS54100	Fall 2020 !	12:00
! 22	CS54100	Spr 2020	13:00



Mapping Relationships

Enforcing REFERENTIAL INTEGRITY

```
CREATE TABLE TEACHES (
    SSN INTEGER,
    CID INTEGER,
    SEMESTER VARCHAR(30),
    TIME DATETIME,
    PRIMARY KEY (SSN, CID, SEMESTER),
    FOREIGN KEY (SSN) REFERENCES PROFESSOR,
    FOREIGN KEY (CID, SEMESTER) REFERENCES COURSE
);
```



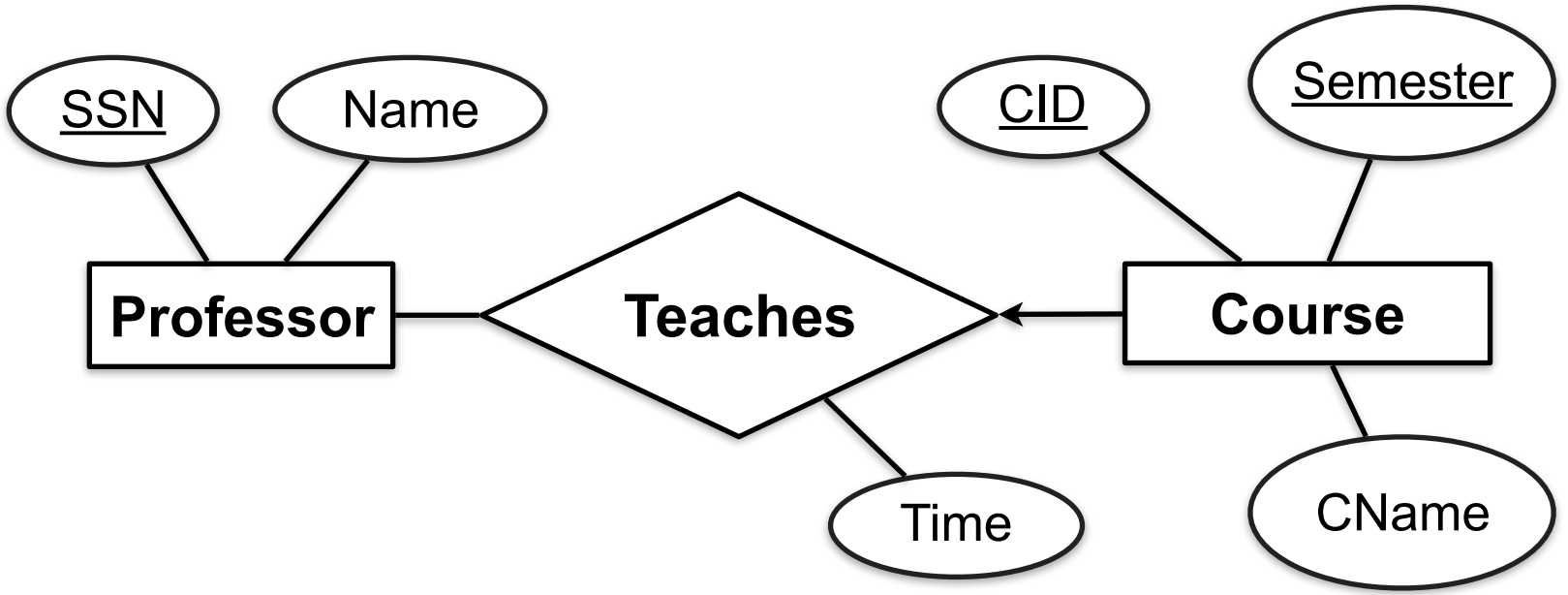
Foreign Keys

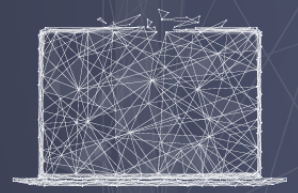
- A Foreign key from relation R to relation S
  - Serves as a pointer or link from a tuple of R to a tuple of S
  - The foreign key must be a primary key for S
  - Any tuple of R can only point to a single tuple of S
  - Multiple tuples in R can point to the same tuple of S
  - This is not a “hard” pointer, but a “logical” pointer — allows the DBMS to move things around on disk or memory
  - The DBMS ensures referential integrity — what about updates / deletes?

Mapping 1-to-Many Relationships

Example 1-to-Many Relationship

What if a course can only have one professor teaching it?

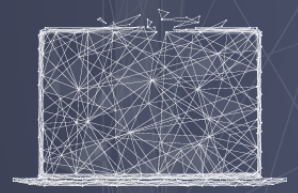




Mapping 1-to-Many Relationships

Two Options

- OPTION 1: Drop SSN from the key of TEACHES
- OPTION 2:
  - Do not create a separate TEACHES relation
  - Instead,
    - include in COURSE a foreign key to PROFESSOR
    - Include the attributes of TEACHES as attributes of COURSE



Mapping 1-to-Many Relationships

Only One Professor Per Course: Option 1

```
CREATE TABLE TEACHES (
    SSN INTEGER,
    CID INTEGER,
    SEMESTER VARCHAR(30),
    TIME DATETIME,
    PRIMARY KEY (CID, SEMESTER),
    FOREIGN KEY (SSN) REFERENCES PROFESSOR,
    FOREIGN KEY (CID, SEMESTER) REFERENCES COURSE
);
```

A given course (CID, Semester) can only appear once in this table — hence can only have one instructor.

Mapping 1-to-Many Relationships

Only One Professor Per Course : Option 2

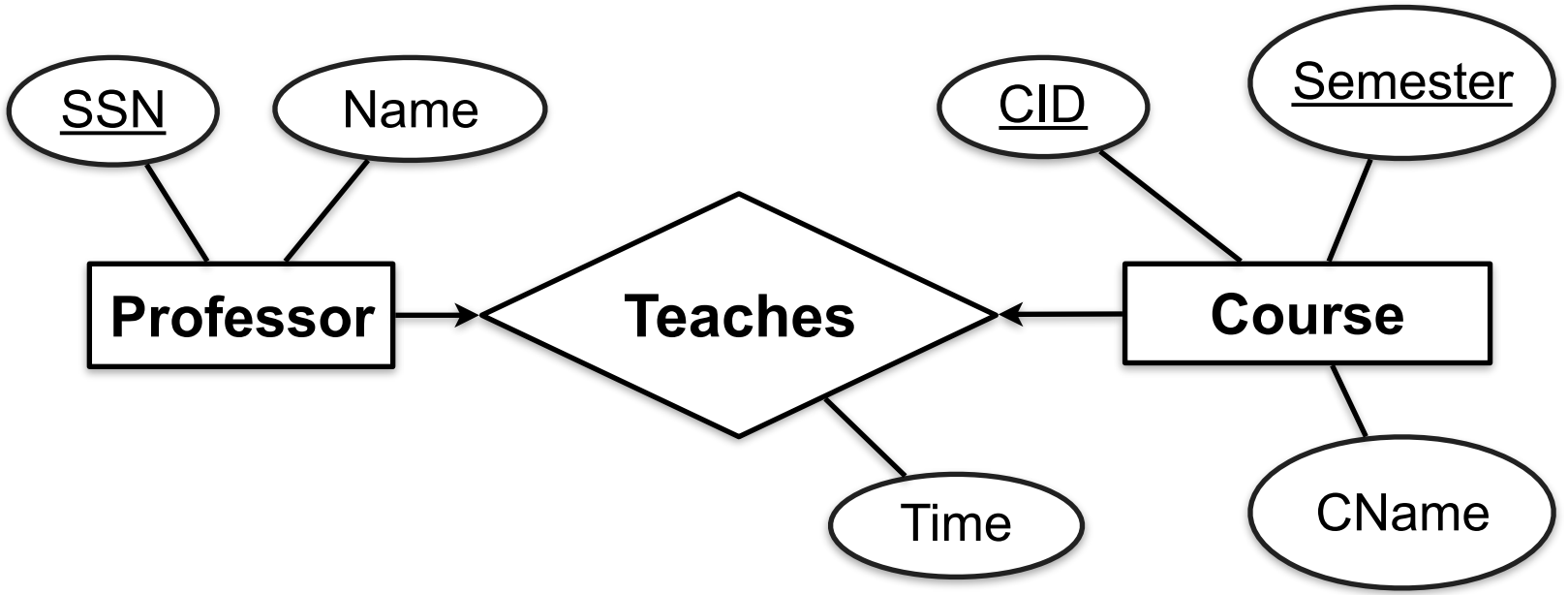
```
CREATE TABLE COURSE (
    CID INTEGER,
    CNAME VARCHAR(30),
    SEMESTER VARCHAR(30),
    INSTRUCTOR INTEGER,
    TIME DATETIME,
    PRIMARY KEY (CID, SEMESTER),
    FOREIGN KEY (INSTRUCTOR)
    REFERENCES PROFESSOR
);
```

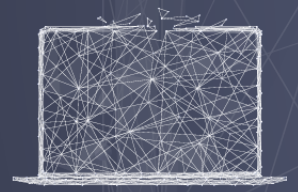
```
CREATE TABLE PROFESSOR (
    SSN INTEGER,
```

Each course appears only once in COURSES, hence can only have one instructor.

Mapping 1-to-1 Relationships

What if a course can only have one professor teaching it, AND each professor can only teach one course?

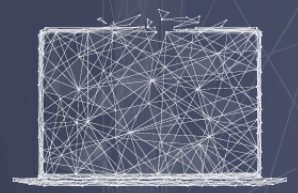




Mapping 1-to-1 Relationships

Three Options

- OPTION 1: Make both SSN and {CID, Semester} keys for TEACHES
- OPTION 2: Don't create a relation for TEACHES, instead
  - Add SSN as an attribute to COURSES as a candidate key
- Option 3: Don't create a relation for TEACHES, instead
  - Add {CID, Semester} to PROFESSOR as a candidate key



Mapping 1-to-1 Relationships

Only One Professor Per Course, Only One Course Per Professor: Option 1

```
CREATE TABLE TEACHES (
    SSN INTEGER,
    CID INTEGER,
    SEMESTER VARCHAR(30),
    TIME DATETIME,
    PRIMARY KEY (CID, SEMESTER),
    UNIQUE (SSN),
    FOREIGN KEY (SSN) REFERENCES PROFESSOR,
    FOREIGN KEY (CID, SEMESTER) REFERENCES COURSE
);
```

Now a professor can only appear once; and a course can only appear once in the teaches table

Mapping 1-to-1 Relationships

Only One Professor Per Course, Only One Course Per Professor: Option 2

```
CREATE TABLE COURSE (
    CID INTEGER,
    CNAME VARCHAR(30),
    SEMESTER VARCHAR(30),
    INSTRUCTOR INTEGER,
    TIME DATETIME,
    PRIMARY KEY (CID, SEMESTER),
    UNIQUE (INSTRUCTOR),
    FOREIGN KEY (INSTRUCTOR)
    REFERENCES PROFESSOR
);
```

```
CREATE TABLE PROFESSOR (
    SSN INTEGER,
    NAME VARCHAR(30),
    PRIMARY KEY (SSN)
);
```

A course can have only one professor, and a professor can only show up once in the COURSE Table

Mapping 1-to-1 Relationships

Only One Professor Per Course, Only One Course Per Professor : Option 3

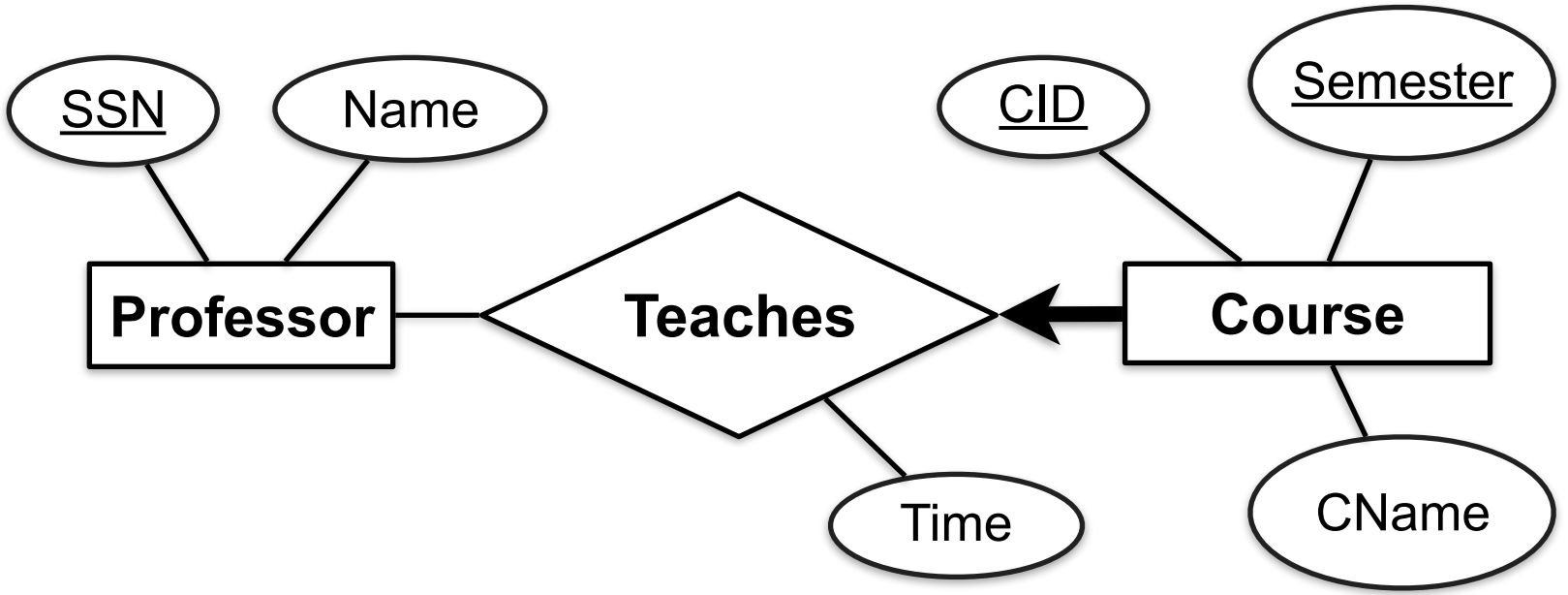
```
CREATE TABLE COURSE (
    CID INTEGER,
    CNAME VARCHAR(30),
    SEMESTER VARCHAR(30),
    PRIMARY KEY (CID,
    SEMESTER),
);
```

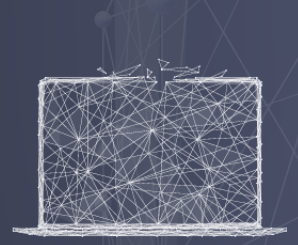
```
CREATE TABLE PROFESSOR (
    SSN INTEGER,
    NAME VARCHAR(30),
    TIME DATETIME,
    CID INTEGER,
    SEMESTER VARCHAR(30),
    PRIMARY KEY (SSN),
    UNIQUE (CID, SEMESTER),
    FOREIGN KEY (CID, SEMESTER)
    REFERENCES COURSE
);
```

We could choose to add the relationship to the PROFESSOR table instead of the COURSE table.

Enforcing Total Participation

What if each course has to have exactly one professor teaching it?





Enforcing Total Participation

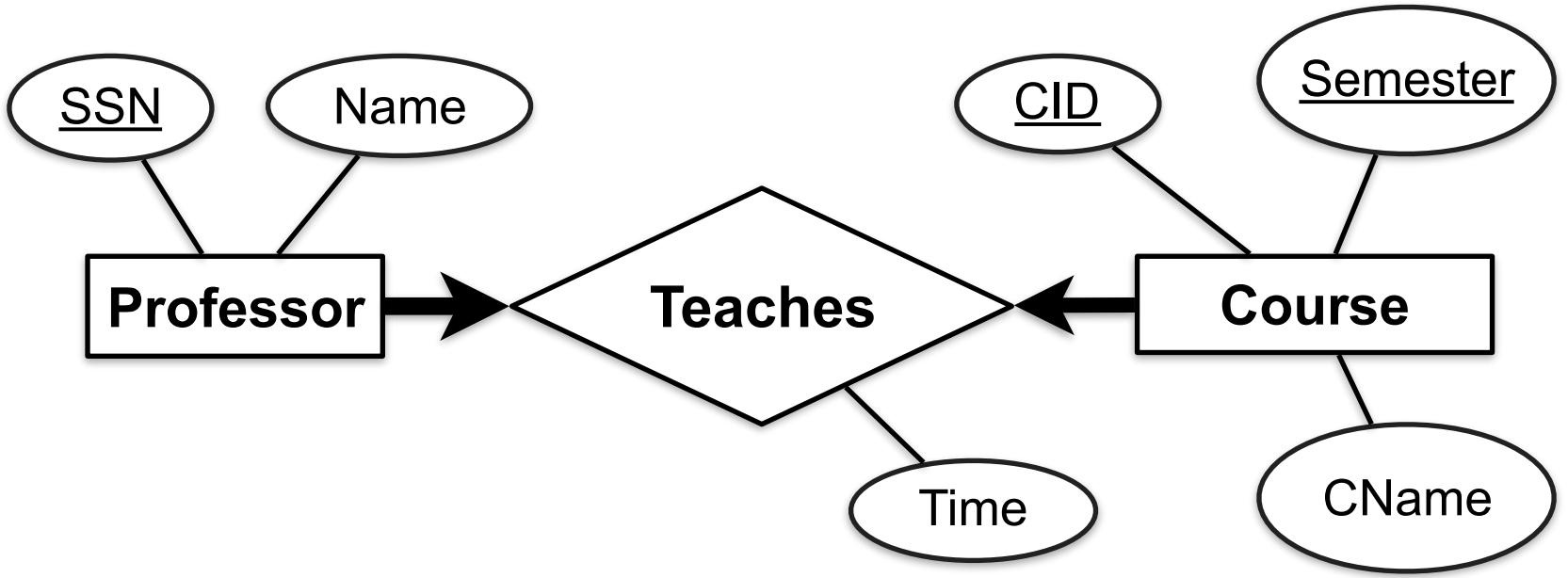
Total Participation of COURSE with 1-to-Many Relationship

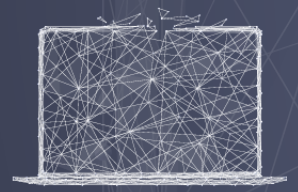
```
CREATE TABLE COURSE (
    CID INTEGER,
    CNAME VARCHAR(30),
    SEMESTER VARCHAR(30),
    INSTRUCTOR INTEGER NOT NULL,
    PRIMARY KEY (CID, SEMESTER),
    FOREIGN KEY (INSTRUCTOR)
    REFERENCES PROFESSOR
);
```

- Add a foreign key in COURSE pointing to PROFESSOR, and do not allow it to be NULL

Enforcing Total Participation

What if each course has to have exactly one professor teaching it, AND every professor must teach exactly one course?



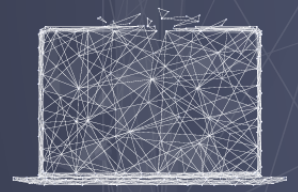


Enforcing Total Participation

Total Participation of COURSE and PROFESSOR with 1-to-1 Relationship

```
CREATE TABLE COURSEPROFESSOR (
    CID INTEGER,
    CNAME VARCHAR(30),
    SEMESTER VARCHAR(30),
    NAME VARCHAR(30),
    TIME DATETIME,
    SSN INTEGER NOT NULL,
    PRIMARY KEY (CID, SEMESTER),
    UNIQUE (SSN)
);
```

- Merge the two entities and the relationship into one relation

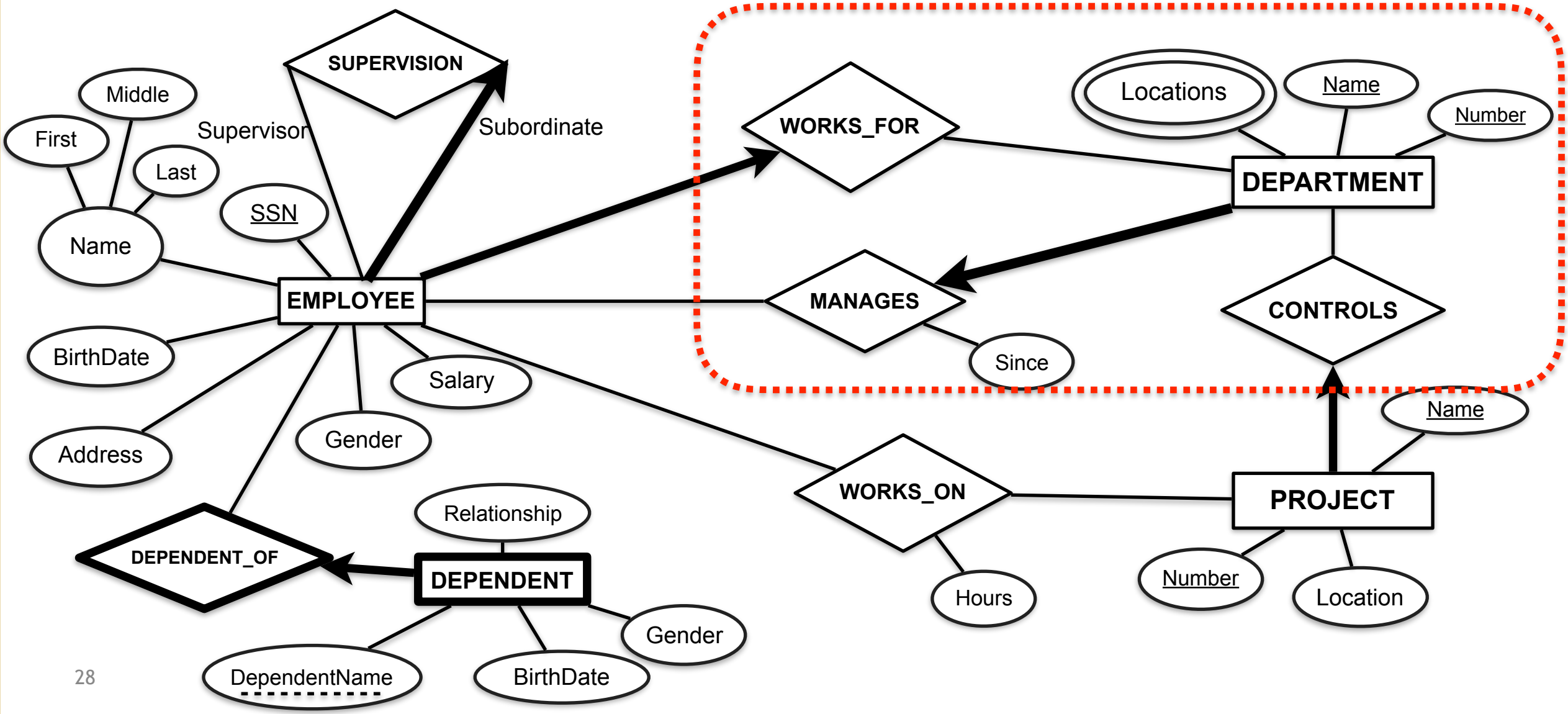


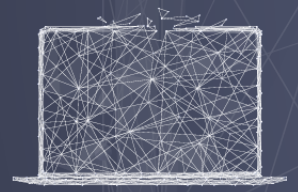
Relationship Constraints

Capturing Cardinality and Participation Constraints

- Through careful design of relations, primary keys, and foreign keys we can enforce several types of participation constraints
- The idea extends to relationships with multiple participating entities
- More generally, we can add additional constraints to relations using the `CHECK` clause

Sample Database ER diagram





MultiValued Attributes

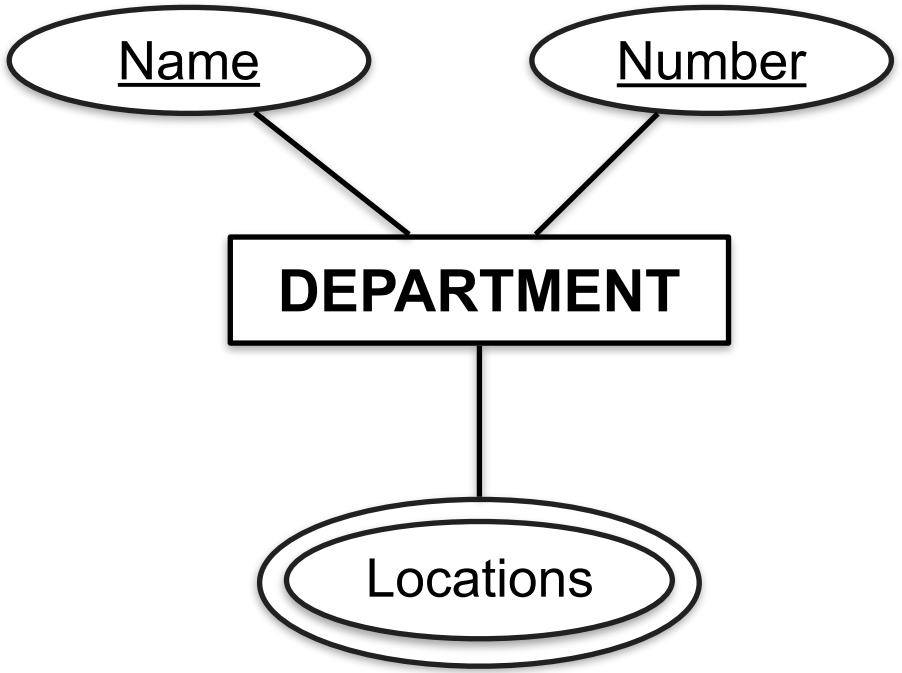
Include the MANAGES Relationship (total and 1:Many Constraint)

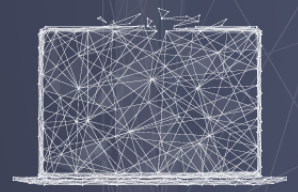
```
CREATE TABLE DEPARTMENT (
    NAME VARCHAR(30),
    NUMBER INTEGER,
    MANAGER INTEGER,
    LOCATIONS VARCHAR(100),
    PRIMARY KEY (NUMBER),
    UNIQUE (NAME),
    FOREIGN KEY (MANAGER)
    REFERENCES EMPLOYEE
);
```

Multivalued Attributes

No Nesting in Relations

- To handle multivalued attributes, we have to create a separate relation for that attribute
- Include a foreign key back to the entity to which this attribute belongs





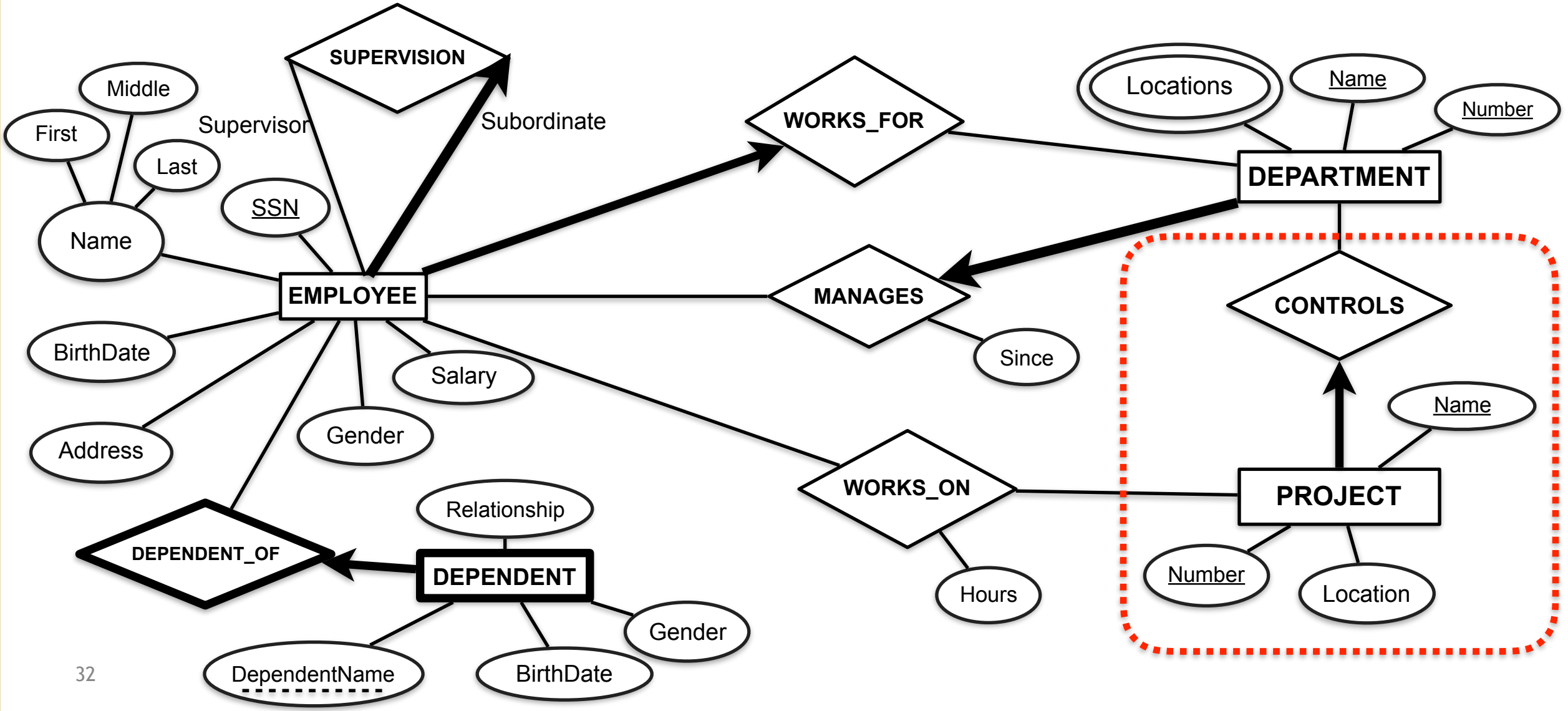
MultiValued Attributes

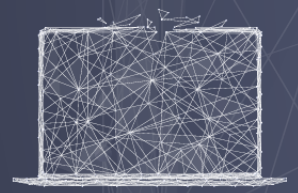
Allowing Multiple Locations for Departments

```
CREATE TABLE DEPARTMENT (
    NAME VARCHAR(30),
    NUMBER INTEGER,
    MANAGER INTEGER,
    PRIMARY KEY (NUMBER),
    UNIQUE (NAME),
    FOREIGN KEY (MANAGER)
    REFERENCES EMPLOYEE
);
```

```
CREATE TABLE DEPT_LOCATION (
    NUMBER INTEGER,
    LOCATION VARCHAR(100),
    PRIMARY KEY (NUMBER, LOCATION),
    FOREIGN KEY (NUMBER)
    REFERENCES DEPARTMENT
);
```

Sample Database ER diagram



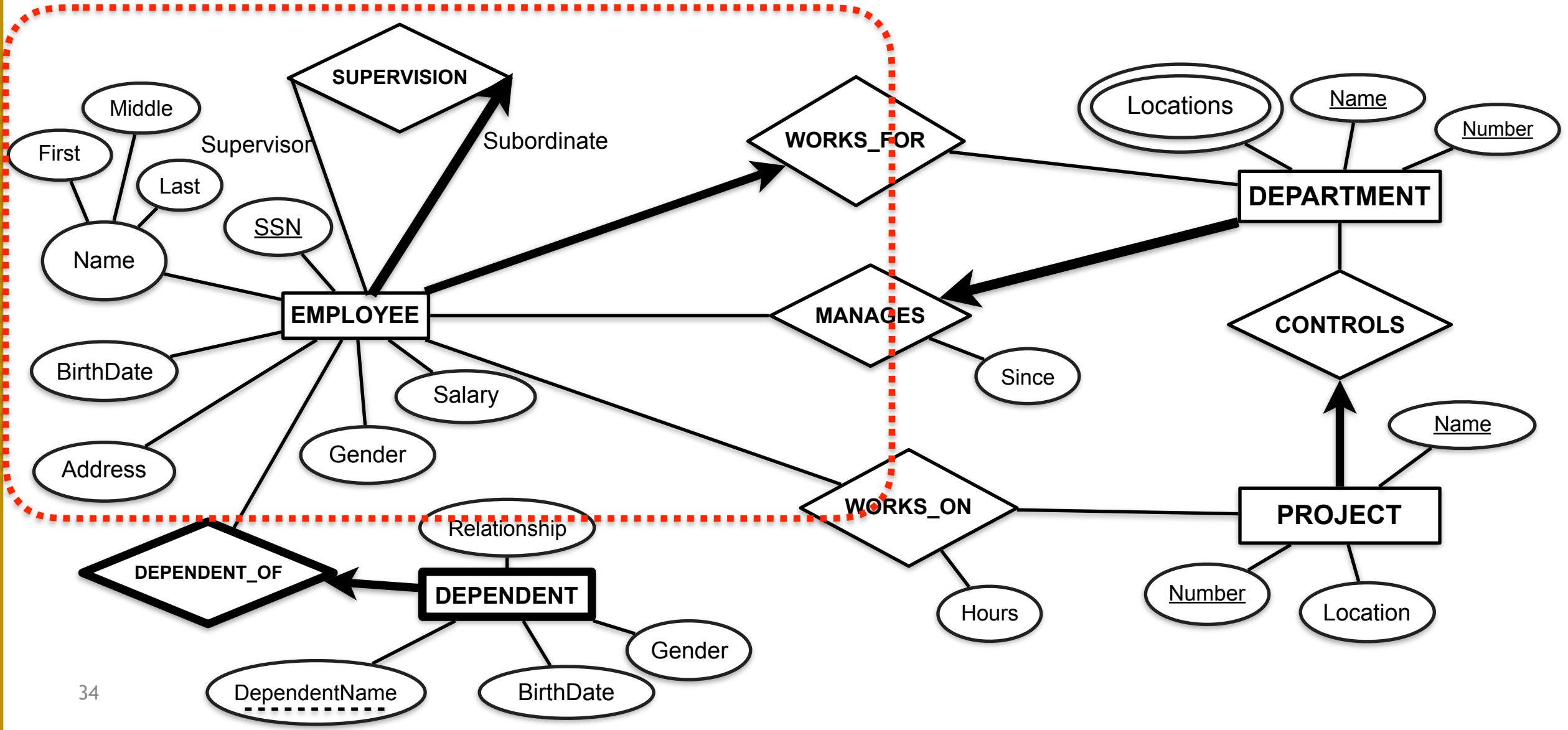


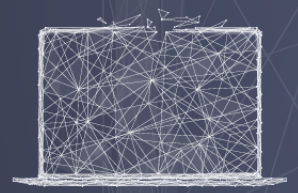
Mapping Strong Entities

The PROJECT Relation

```
CREATE TABLE PROJECT (
    NUMBER INTEGER,
    NAME VARCHAR(30),
    LOCATION VARCHAR(100),
    CONTROLLING_DEPT INTEGER,
    PRIMARY KEY (NUMBER),
    UNIQUE (NAME),
    FOREIGN KEY (CONTROLLING_DEPT) REFERENCES PROJECT
);
```

Sample Database ER diagram

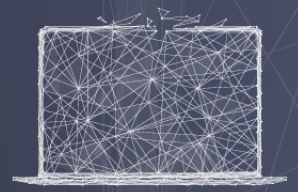




The EMPLOYEE Relation

Include the SUPERVISOR relationships

```
CREATE TABLE EMPLOYEE (
    SSN INTEGER, BirthDate DATETIME, Salary REAL,
    FirstName VARCHAR(30), MiddleName VARCHAR(30),
    LastName VARCHAR(30), Gender VARCHAR(10),
    AddrStreet VARCHAR(30), AddrCity VARCHAR(30),
    AddrState VARCHAR(30), AddrZip NUMBER,
    WORKS_FOR INTEGER, SUPERVISOR INTEGER,
    PRIMARY KEY (SSN)
    FOREIGN KEY (SUPERVISOR) REFERENCES (SSN)
);
```



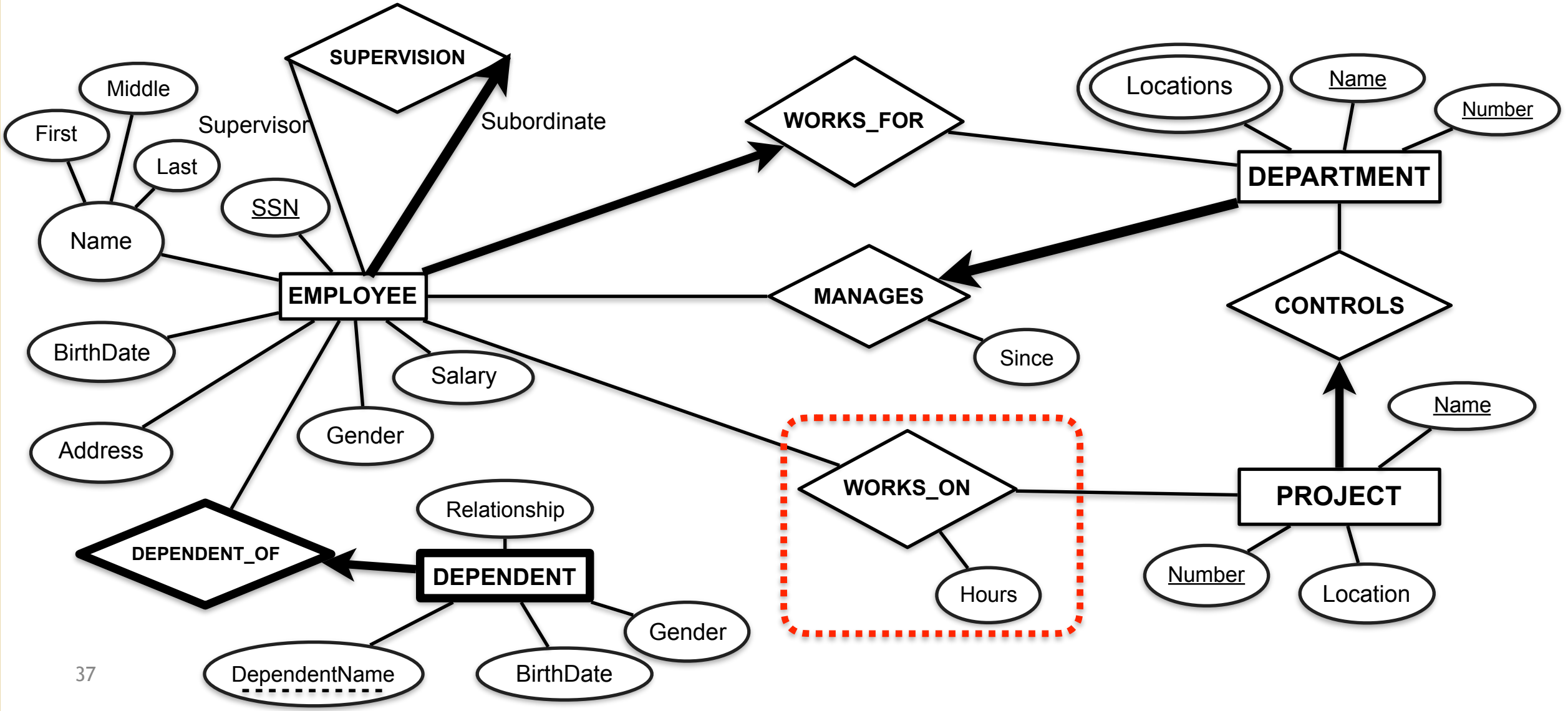
Cyclic Reference

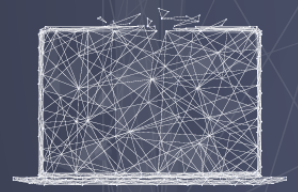
The WORKS_FOR Relationship

- The EMPLOYEE and DEPARTMENT tables need to reference each other
- We can’t do that when creating the first table
- We can use the Alter table command to add it later

```
ALTER TABLE EMPLOYEE ADD FOREIGN KEY
(WORKS_ON) REFERENCES DEPARTMENT;
```

Sample Database ER diagram

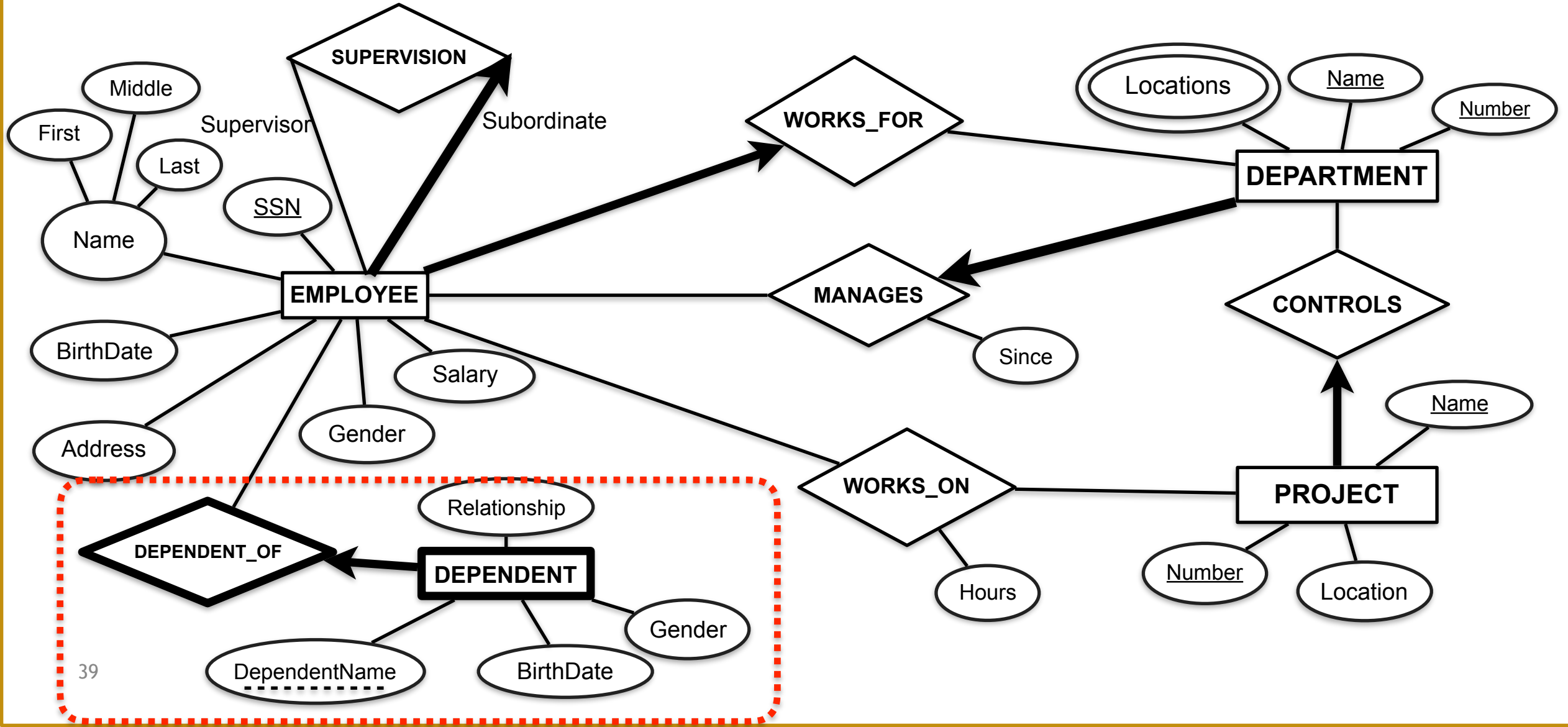


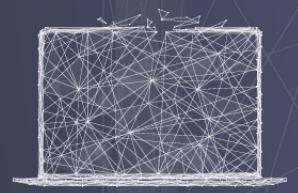


The WORKS_ON Relation

```
CREATE TABLE WORKS_ON (
    SSN INTEGER,
    PROJ_NUMBER INTEGER,
    HOURS INTEGER,
    PRIMARY KEY (SSN, PROJ_NUMBER),
    FOREIGN KEY (SSN) REFERENCES EMPLOYEE,
    FOREIGN KEY (PROJ_NUMBER)  REFERENCES PROJECT
);
```

Sample Database ER diagram

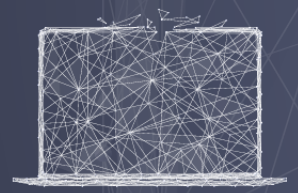




Mapping Weak Entities

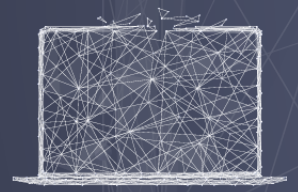
Combine with Defining Relationship

- Create one relation for the weak entity and its defining relationship
- Use the primary key of the owner and the partial key of the weak entity as the key for this relation
- Add all other attributes of the weak entity and the defining relationship.



The DEPENDENTS Weak Entity Relation

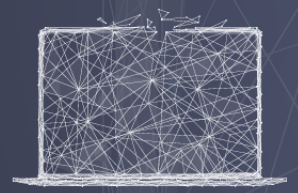
```
CREATE TABLE DEPENDENTS (
    EMPL_SSN INTEGER,
    DEPENDENT_NAME VARCHAR(100),
    RELATIONSHIP VARCHAR(30),
    BIRTDATE DATETIME,
    GENDER VARCHAR(10),
    PRIMARY KEY (EMPL_SSN, DEPENDENT_NAME),
    FOREIGN KEY (EMPL_SSN) REFERENCES EMPLOYEE
);
```



Referential Integrity With Updates

Goal: No Dangling Pointers

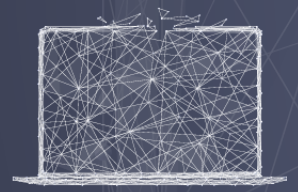
- When inserting data, the DBMS enforces referential integrity
  - We are not allowed to insert an invalid foreign key
- What happens when we delete a tuple to which there are references in other tables? Or change the values of the key attributes?
- Specific actions can be specified:
  - `RESTRICT`: Disallow the deletion (default choice)
  - `CASCADE`: Delete the referencing tuple
  - `SET NULL` or `SET DEFAULT`



The WORKS_ON Relation

```
CREATE TABLE WORKS_ON (
    SSN INTEGER,
    NUMBER INTEGER,
    HOURS INTEGER,
    PRIMARY KEY (SSN, NUMBER),
    FOREIGN KEY (SSN) REFERENCES EMPLOYEE
    ON UPDATE CASCADE ON DELETE CASCADE
    FOREIGN KEY (NUMBER)  REFERENCES PROJECT
    ON UPDATE CASCADE ON DELETE RESTRICT
);
```

PURDUE UNIVERSITY | College of Science

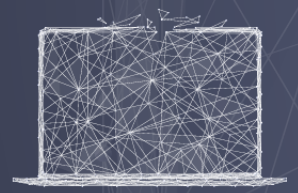


SQL Queries: Defining Structure

Deleting Tables

- Used to remove a relation (table) and its definition.
- The relation can no longer be used in queries, updates, or any other commands

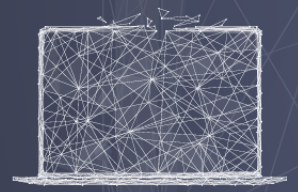
```
DROP TABLE DEPENDENTS;
```



SQL Queries: Defining Structure

Modifying Tables

- The ALTER command can be used to modify an existing table. We can
  - Add / Delete a column
  - Rename / Modify a column
  - Add / Drop / Modify a constraint
  - Rename the Table



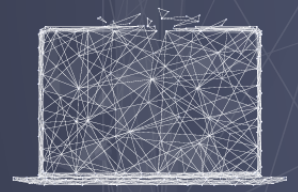
SQL Queries: Defining Structure

Modifying Tables

```
ALTER TABLE EMPLOYEE
ADD COLUMN LOCATION VARCHAR(20);
```

```
ALTER TABLE EMPLOYEE
ADD COLUMN LOCATION VARCHAR(20) DEFAULT 'New York';
```

```
ALTER TABLE EMPLOYEE
DROP COLUMN LOCATION;
```



SQL Queries: Adding Data

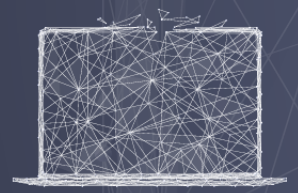
The INSERT Statement

```
INSERT INTO STUDENT(ID, FirstName, LastName, GPA)
VALUES (432432,''Jane'', ''Doe'', 4.0);

INSERT INTO STUDENT(ID, FirstName, LastName, GPA)
VALUES (2343, ''Lee'', ''Keehwan'', 3.8);

INSERT INTO STUDENT(ID, FirstName, LastName, GPA)
VALUES (5434, ''Foster'', ''Ian'', 3.0);
```

```
INSERT INTO STUDENT(ID, FirstName,
LastName, GPA)
VALUES (432432, ''Jane'', ''Doe'', 4.0)
       (2343, ''Lee'', ''Keehwan'', 3.8)
       (5434, ''Foster'', ''Ian'', 3.0);
```



Mapping from ER to Relational Model

Outcome

- In this module we studied how to translate an ER diagram to a corresponding relational database schema
  - Mapping entities and relationships
  - Alternatives choices and their implications
  - How to enforce (some) constraints
  - Using SQL to create, alter or delete a table