

CSU34041

Database Constraints

Yvette Graham
ygraham@tcd.ie



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

Today's Lecture

- Types of Constraints
- Integrity Constraints in Detail
- Constraint Violations
- Complex Constraints
- Task 5 - hand up before class ends



Integrity vs Security



Integrity and **Security** are related but they are not the same

- Integrity is concerned with accidental corruption
- Security is concerned with deliberate corruption

Integrity

- Integrity Constraints

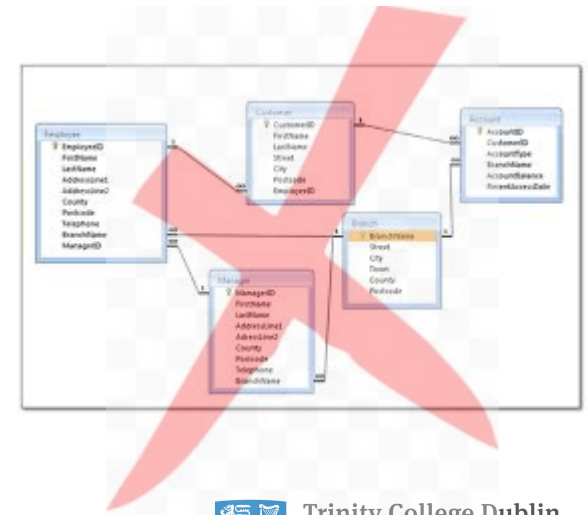
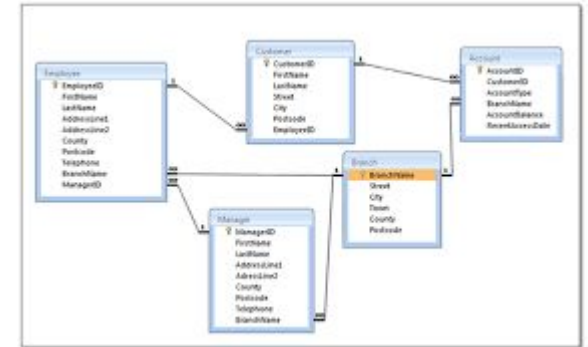
Security

- Security Policies
- Access Control



Relational Model Constraints

- Constraints expressed within the Relational Schema
 - Explicit Constraints
- Constraints that cannot be expressed within the Relational Schema
 - Semantic Constraints
 - can be expressed in SQL in some cases
 - usually enforced by the application programs



Integrity Constraints



- Three types of integrity constraint are considered part of the relational model:
 - Key
 - Entity Integrity
 - Referential Integrity
- The **DBMS** must be able to enforce these constraints



Key Constraints



- Specifies that there **may not be any duplicate entries** in key attributes
 - Primary Key
 - Candidate Keys
- Keys are used to uniquely identify a tuple
 - Having a duplicate value in a Key implies that we cannot uniquely identify some tuples



Entity Integrity Constraints

- Specifies that there **may not be any NULL values** in the Primary Key attribute
 - The Primary Key is used to uniquely identify each tuple in a relation
 - Having NULL in a Primary Key implies that we cannot identify some tuples



Referential Integrity

- **Entity** constraints are specified on individual relations
- **Referential** Integrity constraints are specified between two relations
 - Maintains consistency among tuples in the two relations
 - A tuple in one relation that refers to another relation, must refer to an existing tuple in that relation
- A Foreign Key formally specifies a Referential Integrity Constraint between two relations



NULL Keys

- As per the Entity Integrity constraint
 - No part of a Primary Key can be NULL
- However, Foreign Keys in certain circumstances may be NULL
 - A decision must be made during schema design as to whether it is valid for the foreign key to be NULL at any point



NULL Key Example



NULL Key Example

Can GP_ID be NULL?



NULL Key Example

NO! GP_ID is a primary key



NULL Key Example

Can GP be NULL?



NULL Key Example

Yes! GP is only a foreign key so it can be NULL



Referential Integrity

- When defining an attribute as a Foreign Key
 - You must also specify whether or not the foreign key is allowed to contain NULLs
- In the case of a composite Foreign Key
 - if the Foreign Key is allowed to contain NULLs then either *all the component attributes* should be NULL or *none of them* NULL
 - in order to enforce referential integrity



NULL Keys

STUDENT

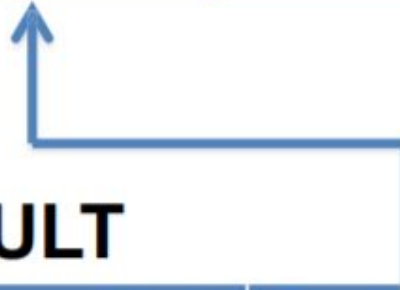
<u>Student_No</u>	Name	Address
-------------------	------	---------

RESULT

<u>Course_Code</u>	<u>Student_No</u>	Grade
--------------------	-------------------	-------

COURSE

<u>Course_Code</u>	Title	Lecturer
--------------------	-------	----------



NULL Key

Question: should we allow student_no in result table to be NULL?

STUDENT

<u>Student_No</u>	Name	Address
-------------------	------	---------

RESULT

<u>Course_Code</u>	<u>Student_No</u>	Grade
--------------------	-------------------	-------

COURSE

<u>Course_Code</u>	Title	Lecturer
--------------------	-------	----------



NULL Key

No! this would defy logic
- a result should not be
able to exist without a
student

STUDENT

<u>Student_No</u>	Name	Address
-------------------	------	---------

RESULT

<u>Course_Code</u>	<u>Student_No</u>	Grade
--------------------	-------------------	-------

COURSE

<u>Course_Code</u>	Title	Lecturer
--------------------	-------	----------



Constraint Violation

There are three basic operations that modify the state of relations in a DB

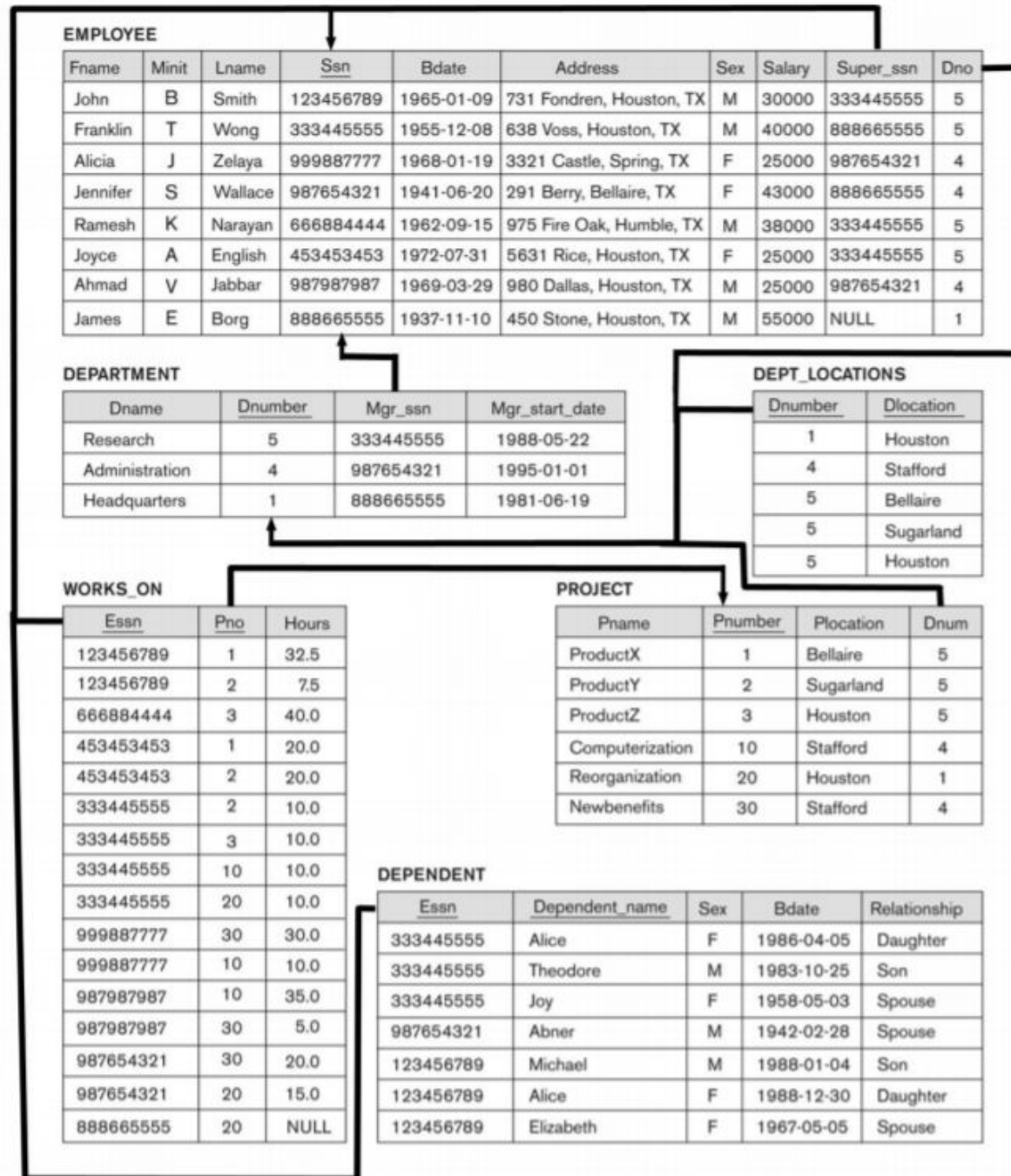
- Insert
- Update
- Delete

These operations should not violate the integrity constraints specified for the DB – Key, Entity, Referential



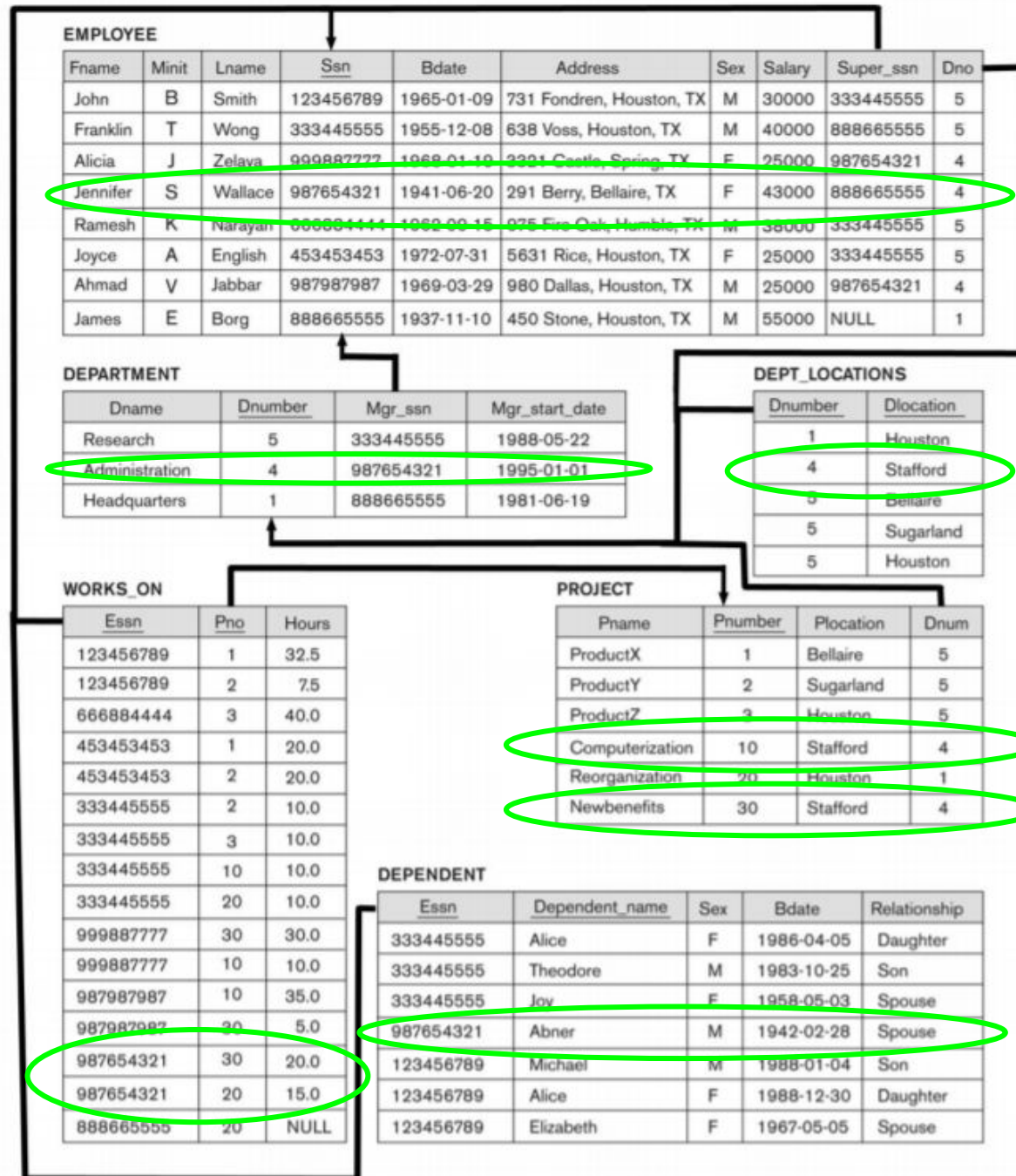
Example

- Employee
- Department
- Dept Location
- Project
- Works on
- Dependent



Example

- Employee
- Department
- Dept Location
- Project
- Works on
- Dependent



Insert Constraint Violation

Insert provides a list of attribute values for a **new tuple t** that is to be added to relation R

- Inserts can violate all the integrity constraints that we have discussed
 - Key
 - Entity Integrity
 - Referential Integrity



Insert Constraint Violation

*Key constraints - **may not be any duplicate entries** in key attributes*

tuple t that is to be

constraints that we have

discussed

- Key
- Entity Integrity
- Referential Integrity



Insert Constraint Violation

Key constraints
entries

Entity constraints - **may not be any NULL values** in the primary Key attribute

tuple t that is to be

constraints that we have

discussed

- Key
- Entity Integrity
- Referential Integrity



Insert Constraint Violation

Key constraints

Entity constraints

Example: a **Foreign Key** formally specifies a referential integrity constraint between two relations

tuple t that is to be

we have

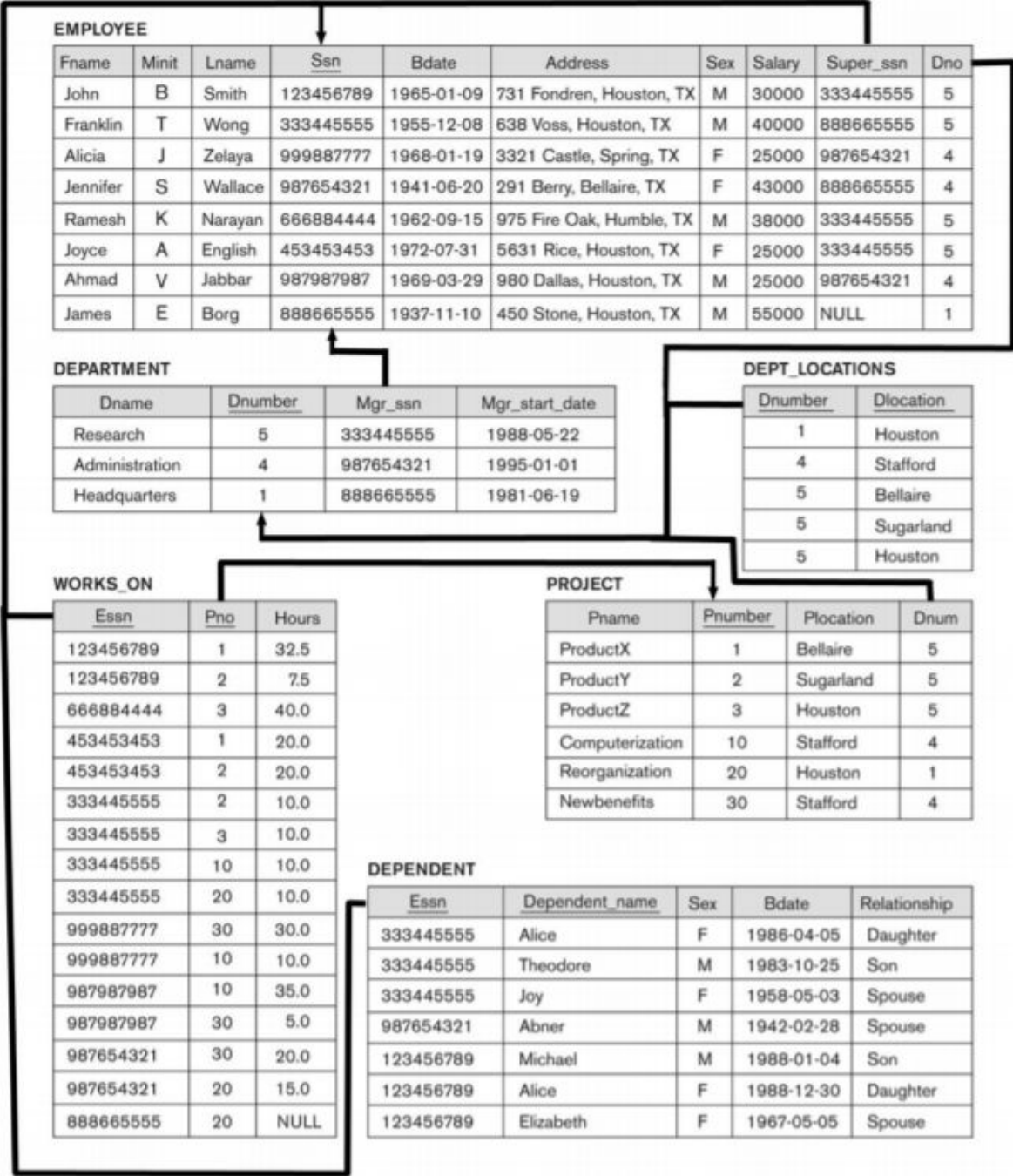
discussed

- Key
- Entity Integrity
- Referential Integrity



Can we insert the following into EMPLOYEE?

<‘Cecilia’, ‘F’, ‘Kolonsky’, NULL, ‘1960-04-05’,
‘6357 Windy Lane, Katy, TX’, F, 28000, NULL, 4>

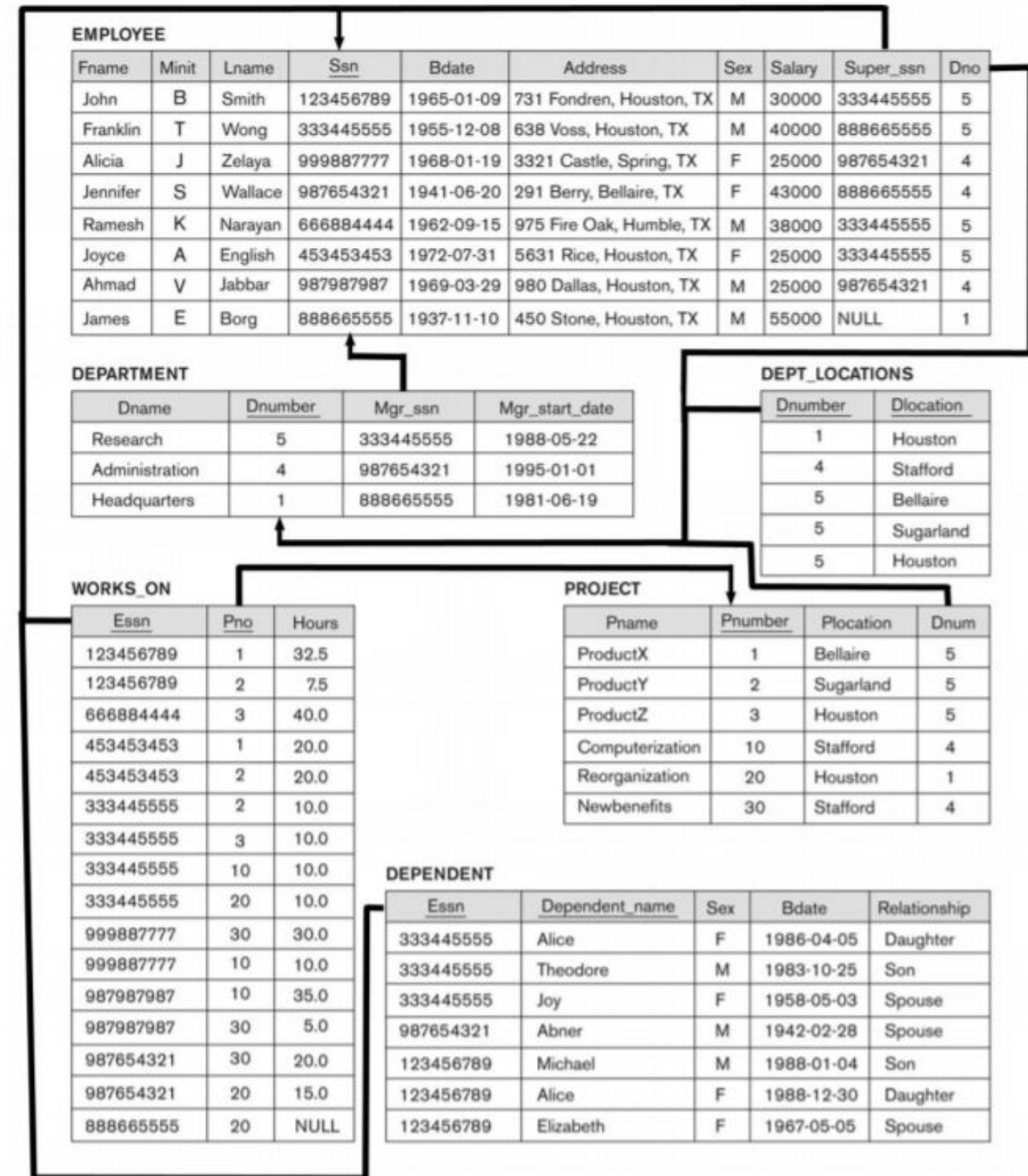


Can we insert the following into EMPLOYEE?

<'Cecilia', 'F', 'Kolonsky', **NULL**, '1960-04-05',
'6357 Windy Lane, Katy, TX', F, 28000, NULL, 4>

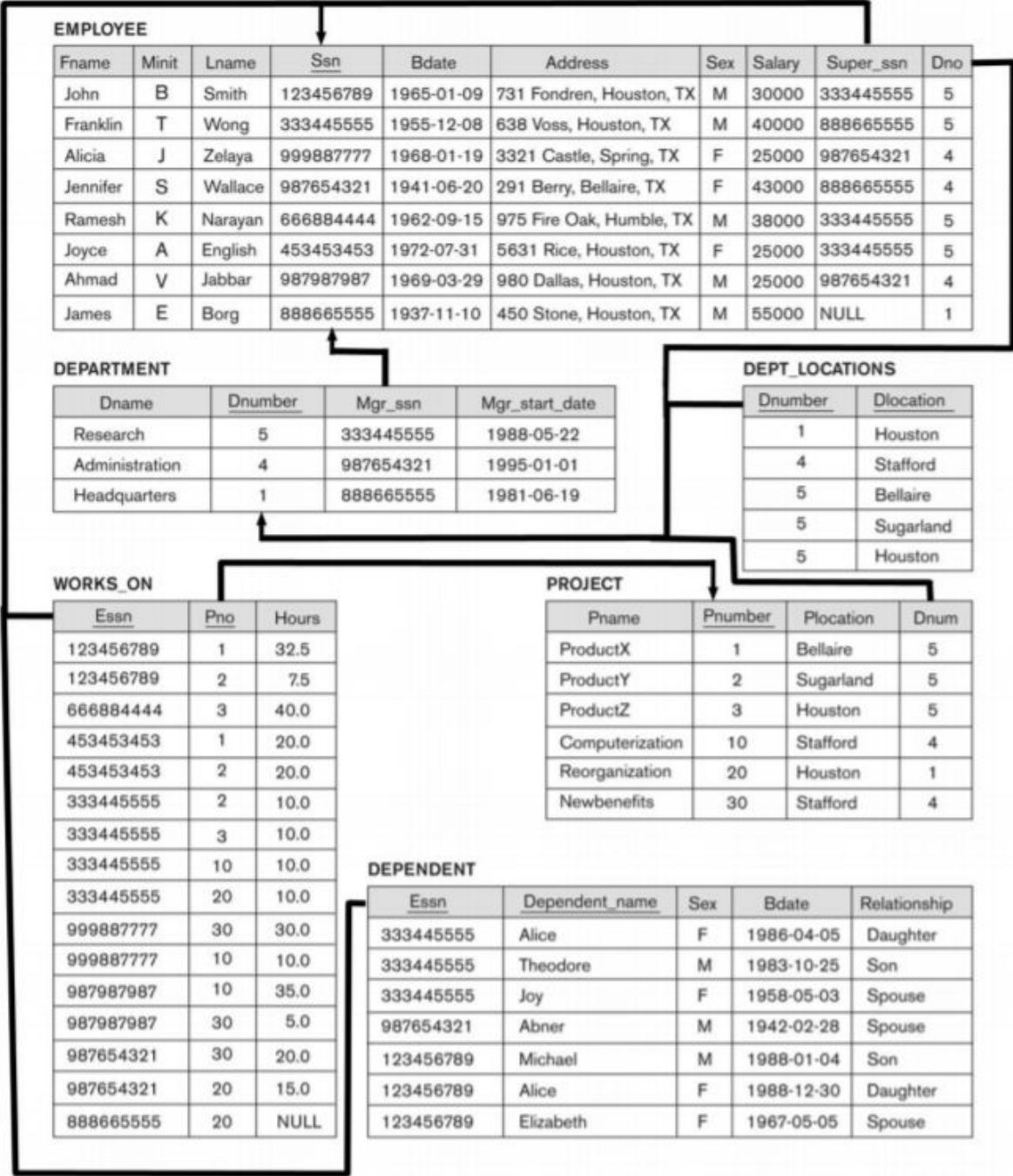
Answer:

- No, this insertion violates the **Entity Integrity** constraint
- NULL is provided for the Primary Key, Ssn, so the insertion is rejected.



Can we insert the following into EMPLOYEE?

<‘Alicia’, ‘J’, ‘Zelaya’, ‘999887777’, ‘1960-04-05’,
‘6357 Windy Lane, Katy, TX’, F, 28000,
‘987654321’, 4>

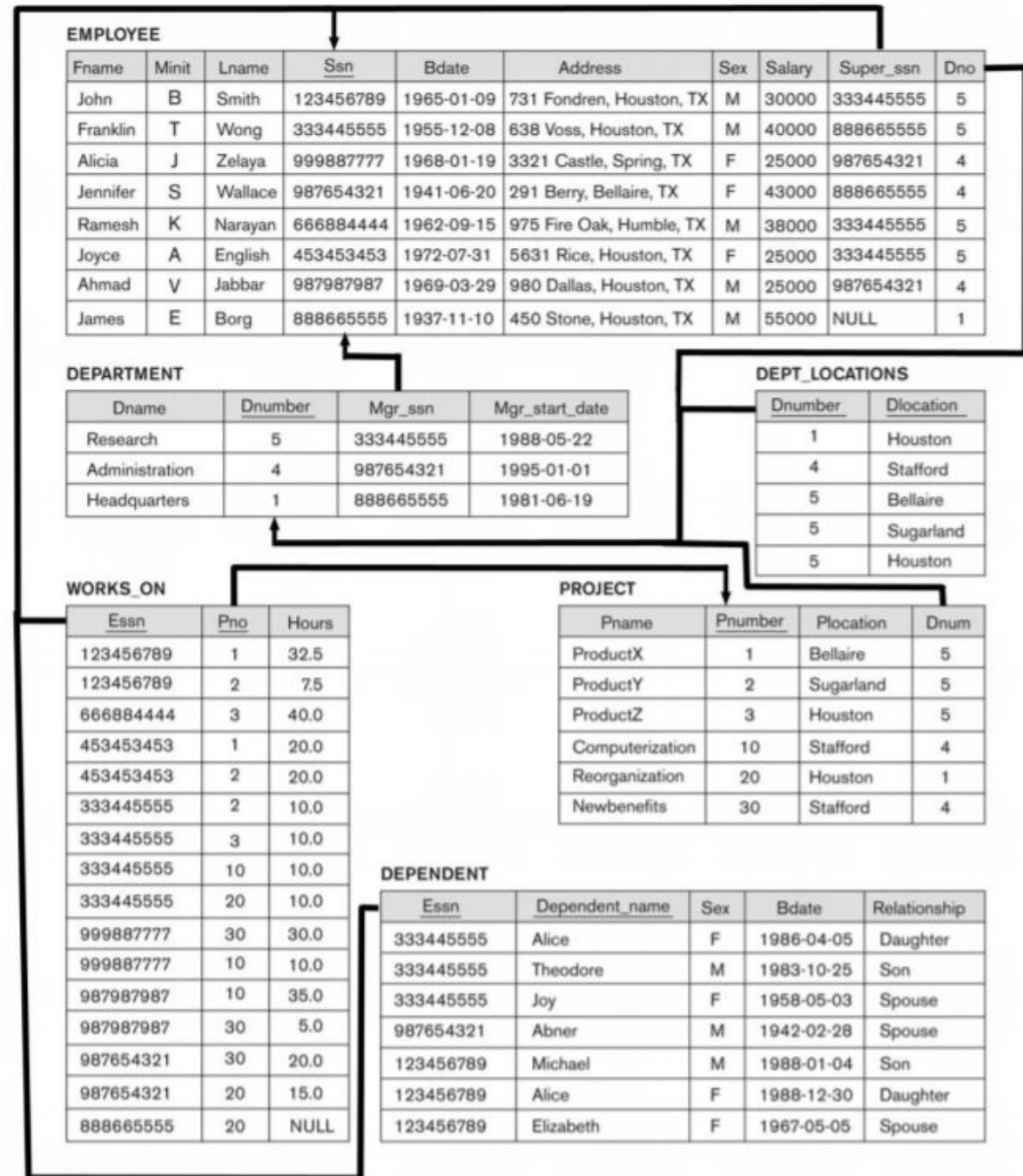


Can we insert the following into EMPLOYEE?

<'Alicia', 'J', 'Zelaya', '999887777', '1960-04-05',
'6357 Windy Lane, Katy, TX', F, 28000,
'987654321', 4>

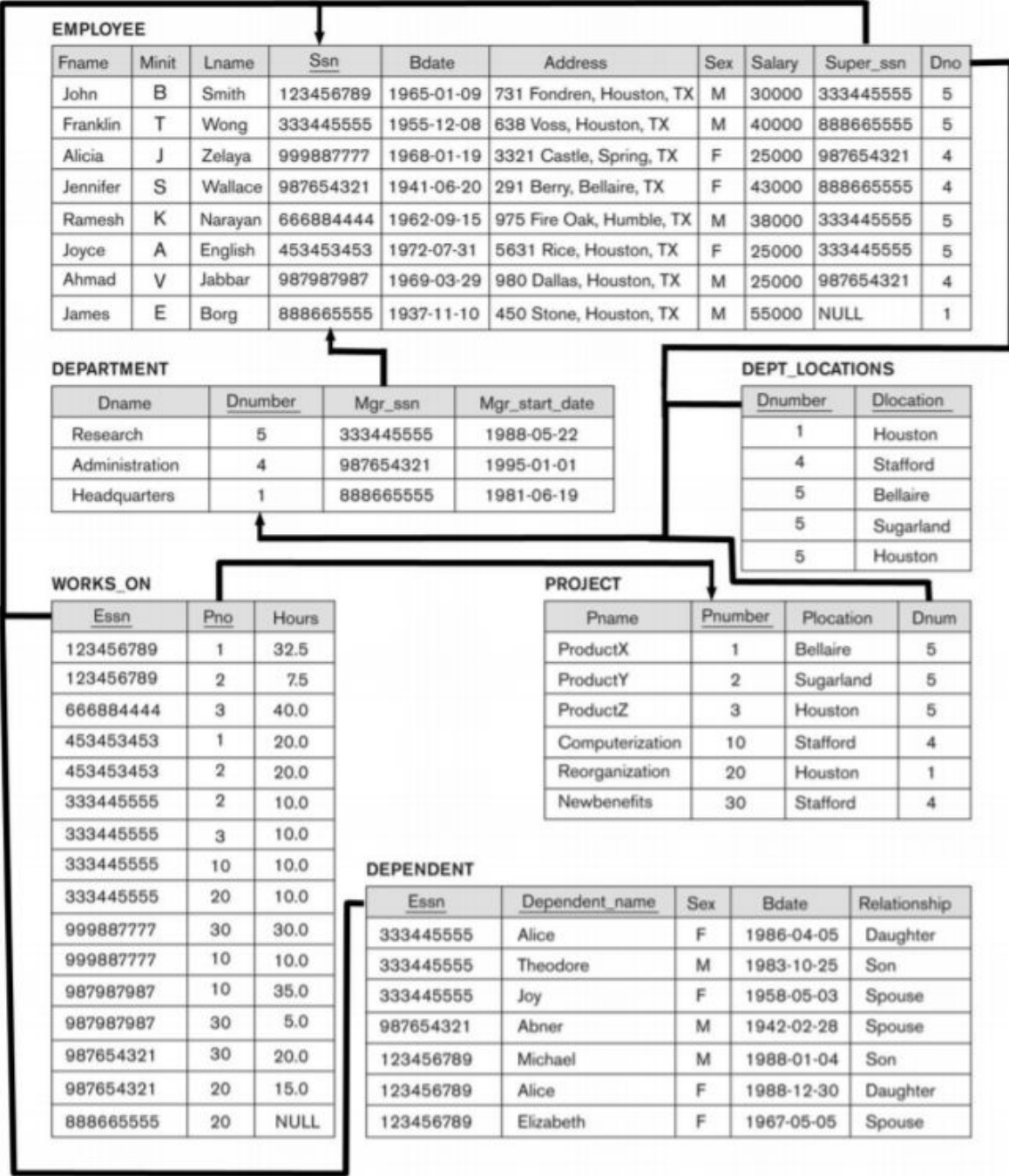
Answer:

- No, this insertion violates the **Key constraint**
- Another tuple with the same Ssn value already exists in the EMPLOYEE relation, so it is rejected



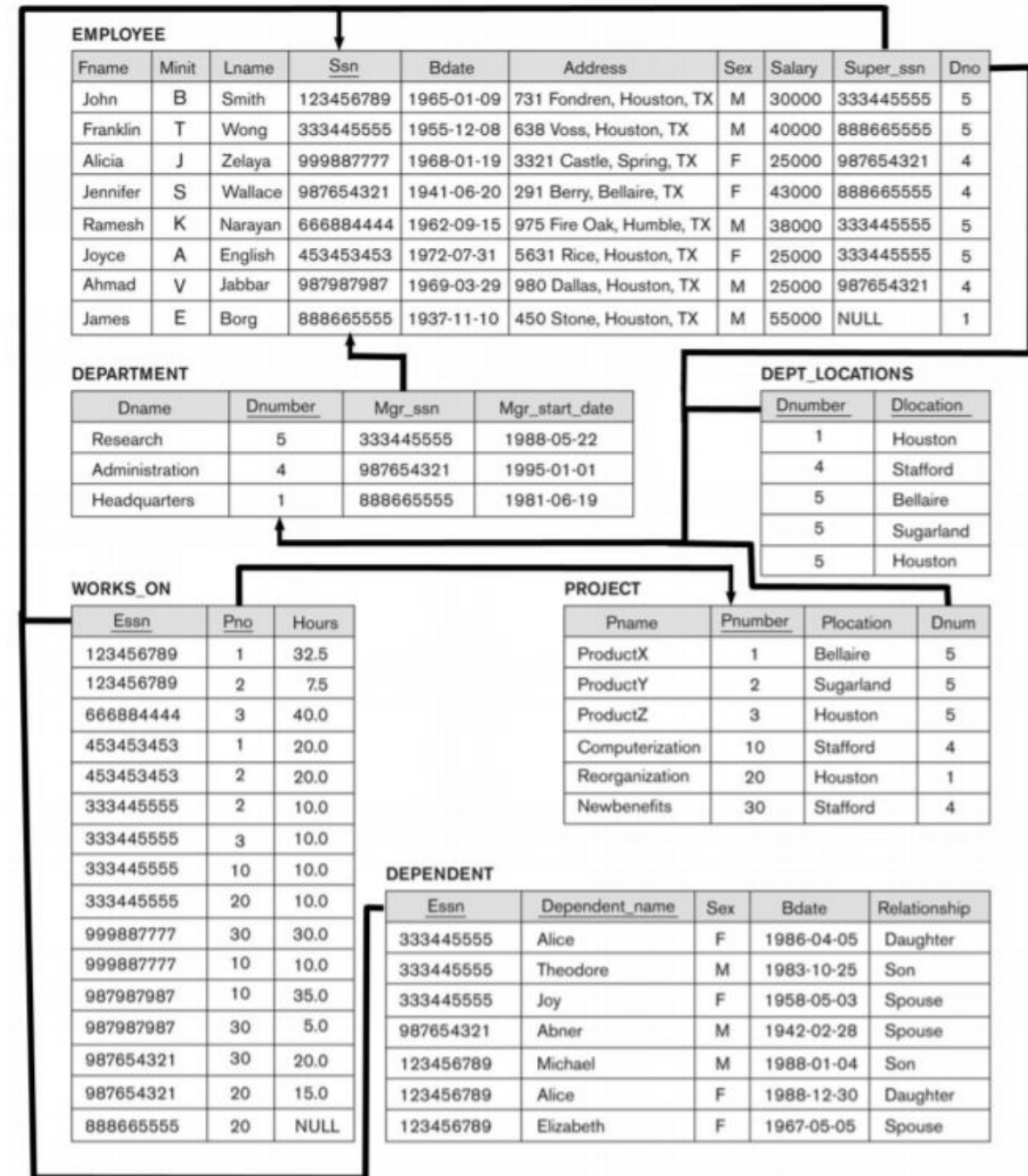
Can we insert the following into EMPLOYEE?

<‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’,
‘1960-04-05’, ‘6357 Windswept, Katy, TX’, F,
28000, ‘987654321’, 7>



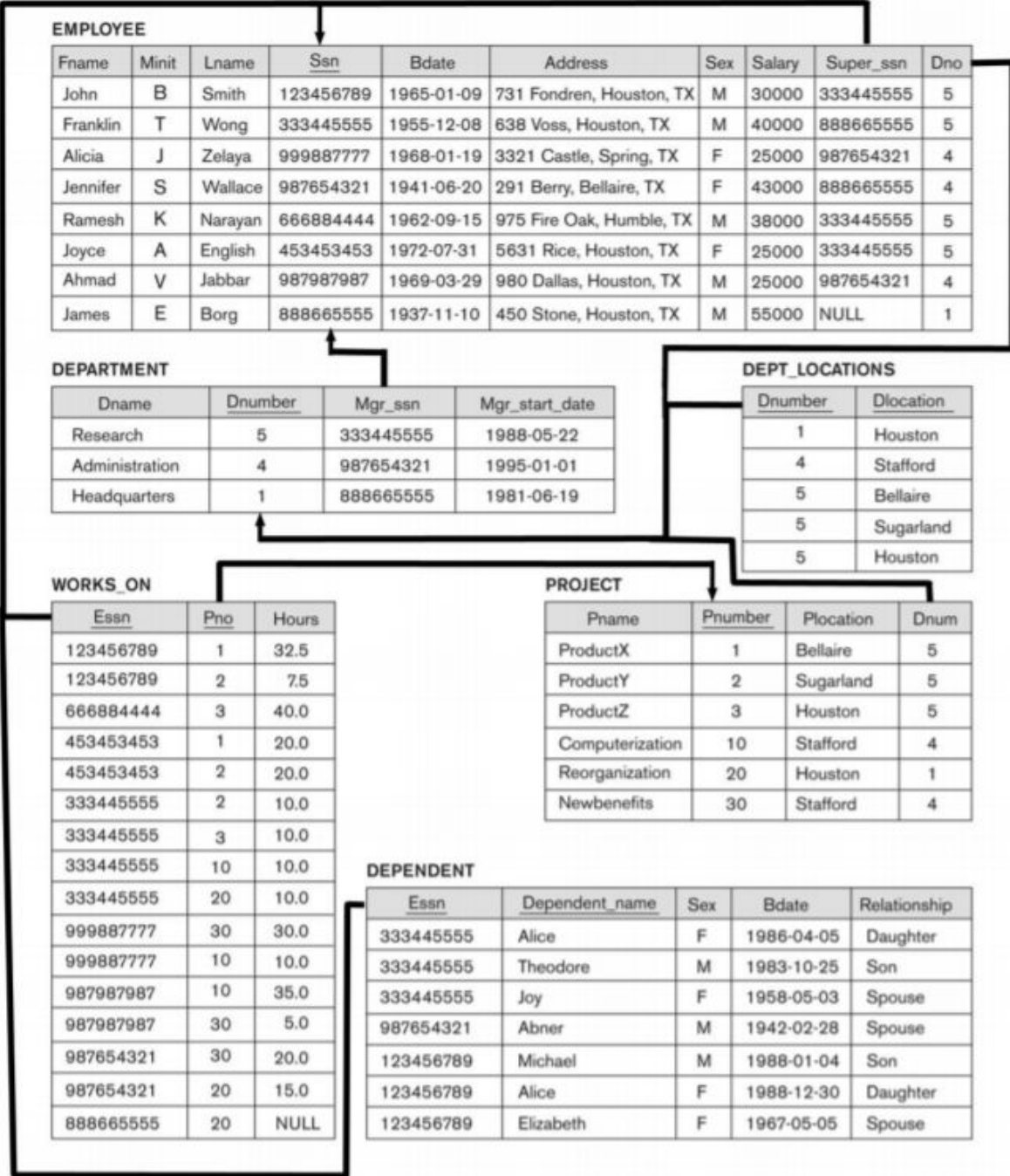

```
<'Cecilia', 'F', 'Kolonsky', '677678989',  
'1960-04-05', '6357 Windswept, Katy, TX', F,  
28000, '987654321', 7>
```

- No, this insertion violates the **referential integrity constraint** specified on Dno in EMPLOYEE
- No corresponding tuple exists in DEPARTMENT with Dnumber = 7



Can we insert the following into EMPLOYEE?

<‘Cecilia’, ‘F’, ‘Kolonsky’, ‘677678989’,
‘1960-04-05’, ‘6357 Windy Lane, Katy, TX’, F,
28000, NULL, 4>



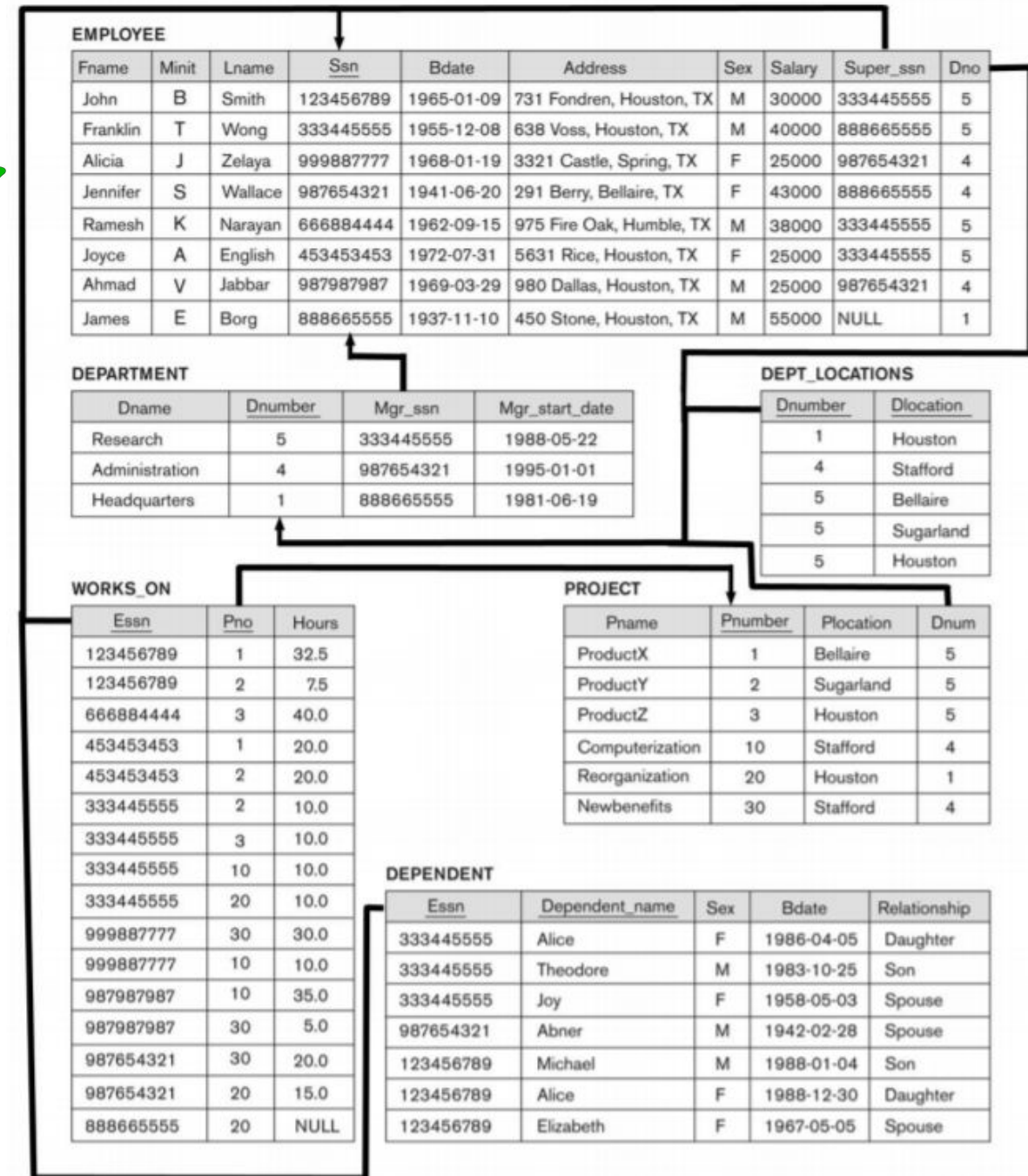
Can we insert the following into EMPLOYEE?

<'Cecilia', 'F', 'Kolonsky', '677678989',
'1960-04-05', '6357 Windy Lane, Katy, TX', F,
28000, NULL, 4>



Answer:

- Yes, this insertion satisfies all constraints, so it is acceptable



Constraint Violation - Delete

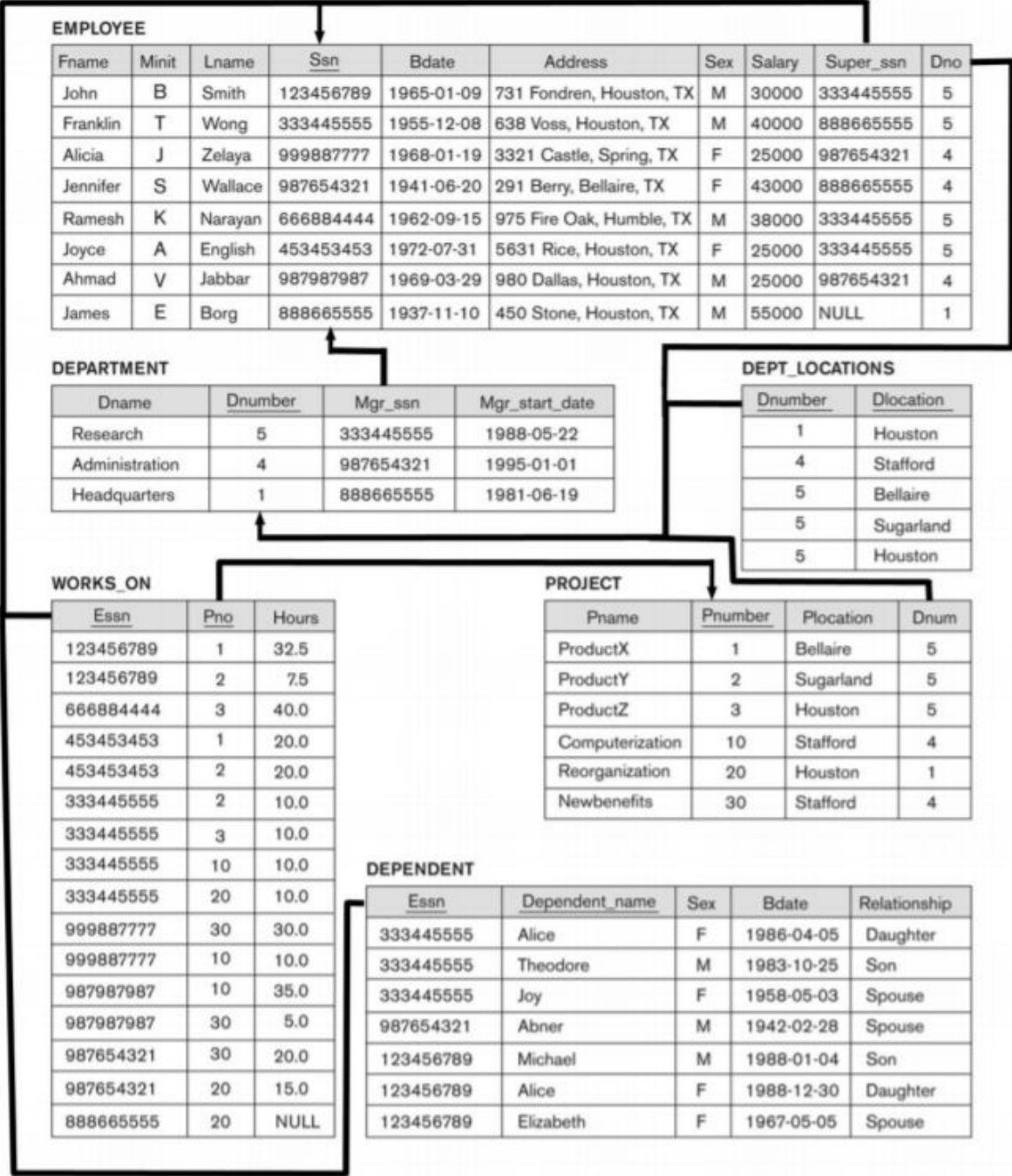
To specify a deletion, a condition on the attributes of a relation is created which selects one or more tuples to be deleted

- The Delete operation can only violate the Referential Integrity constraint



Can we delete the following?

Any tuples in WORKS_ON with Essn = '999887777' and Pno = 10



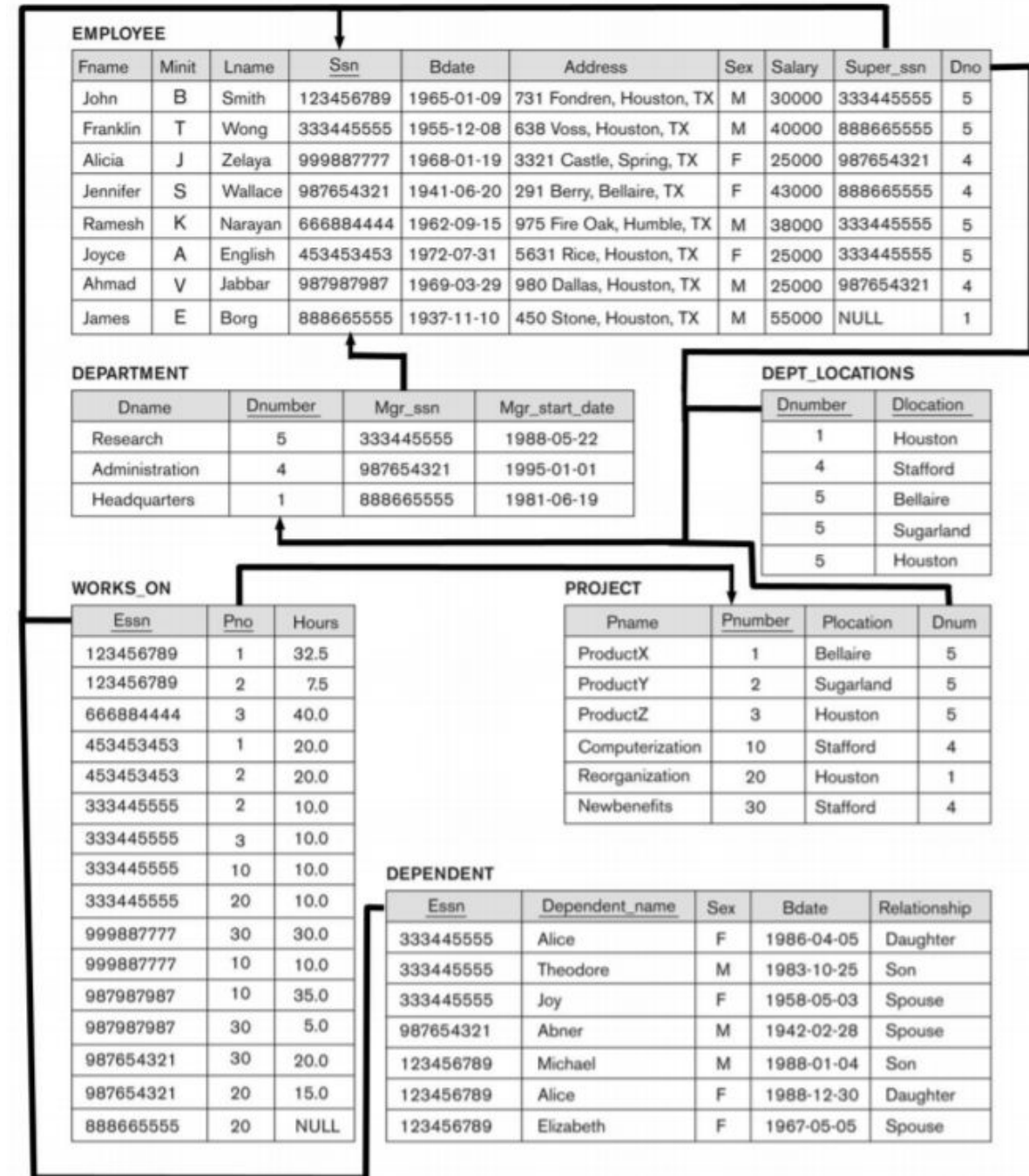
Can we delete the following?

Any tuples in WORKS_ON with Essn = '999887777' and Pno = 10



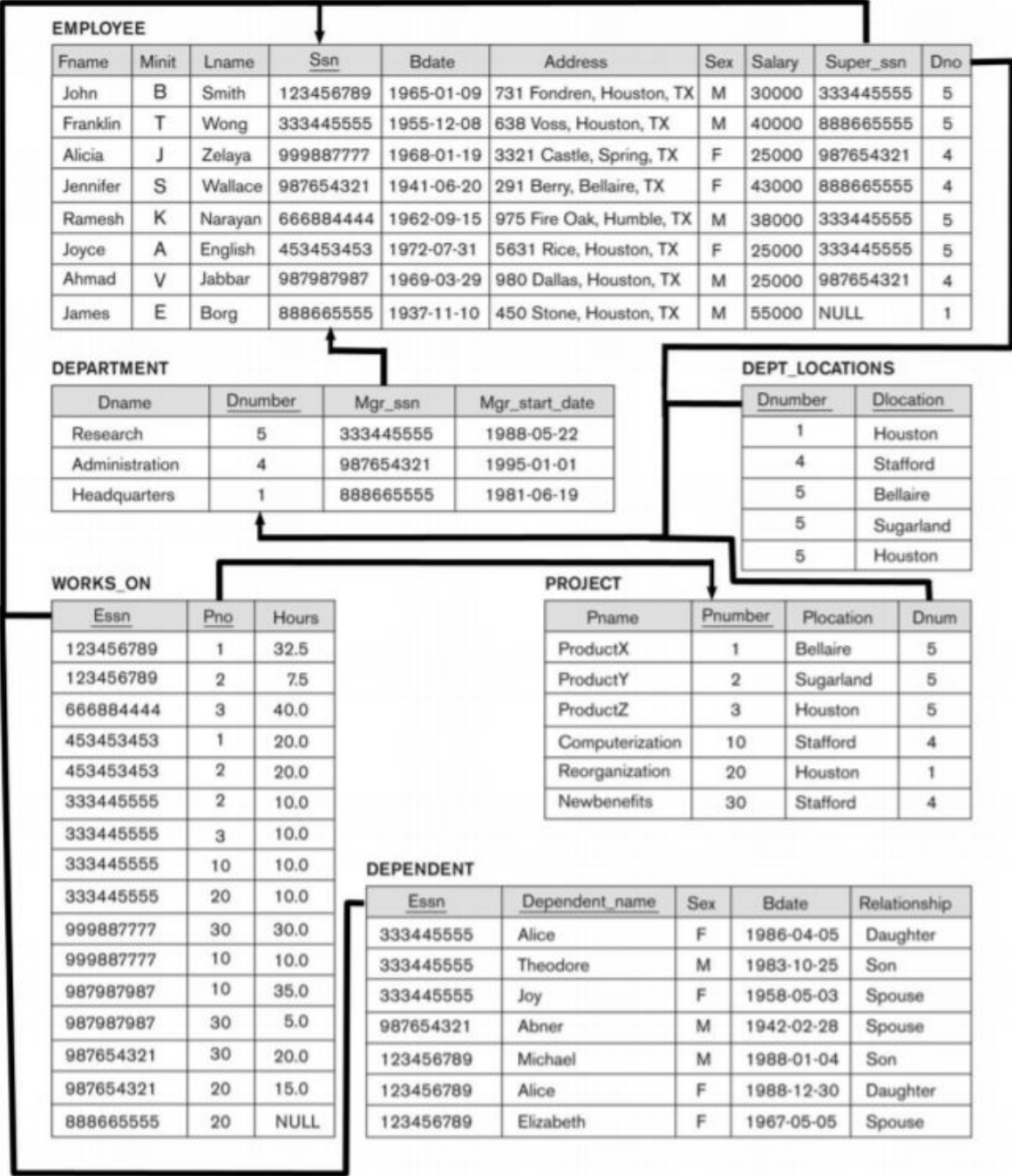
Answer:

- Yes, this deletion is acceptable and deletes exactly one tuple



Can we delete the following?

Any tuples in EMPLOYEE with Ssn = '999887777'

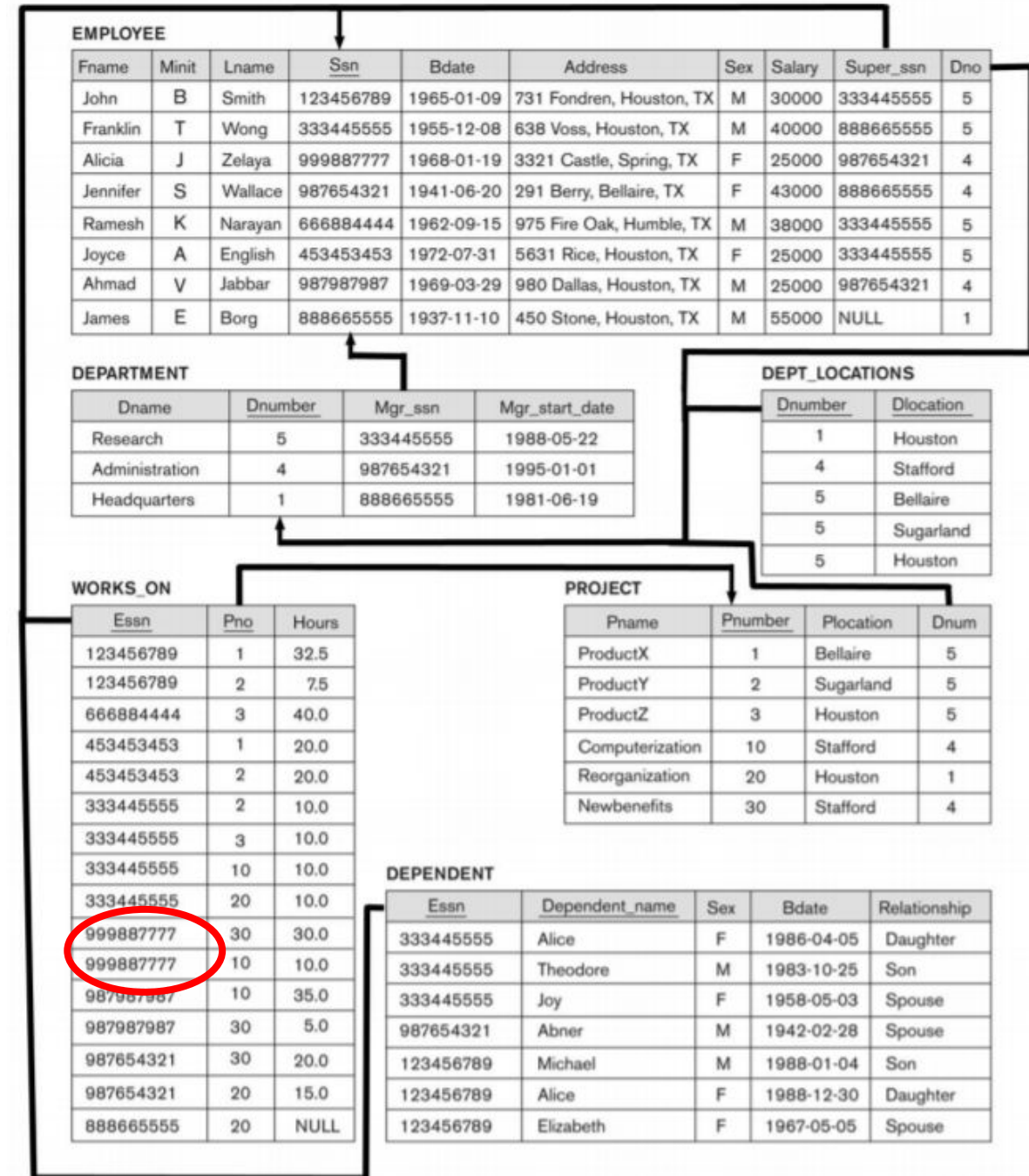


Can we delete the following?

Any tuples in EMPLOYEE with Ssn = '999887777'

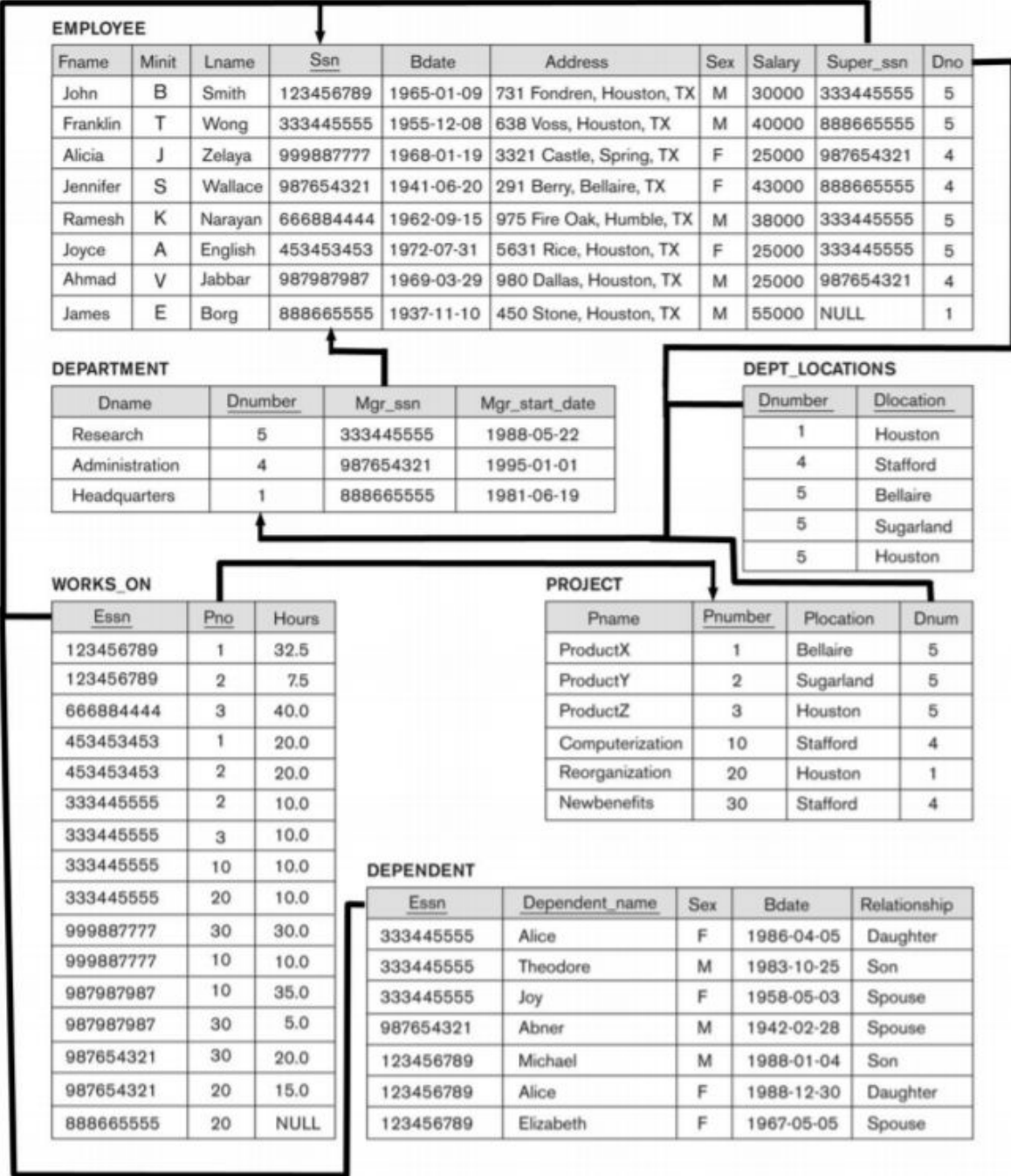
Answer:

- No, this deletion is not acceptable, because there are tuples in WORKS_ON that refer to this tuple
- Hence, if the tuple in EMPLOYEE is deleted, Referential Integrity violations will result



Can we delete the following?

Any tuples in EMPLOYEE with Ssn = '333445555'

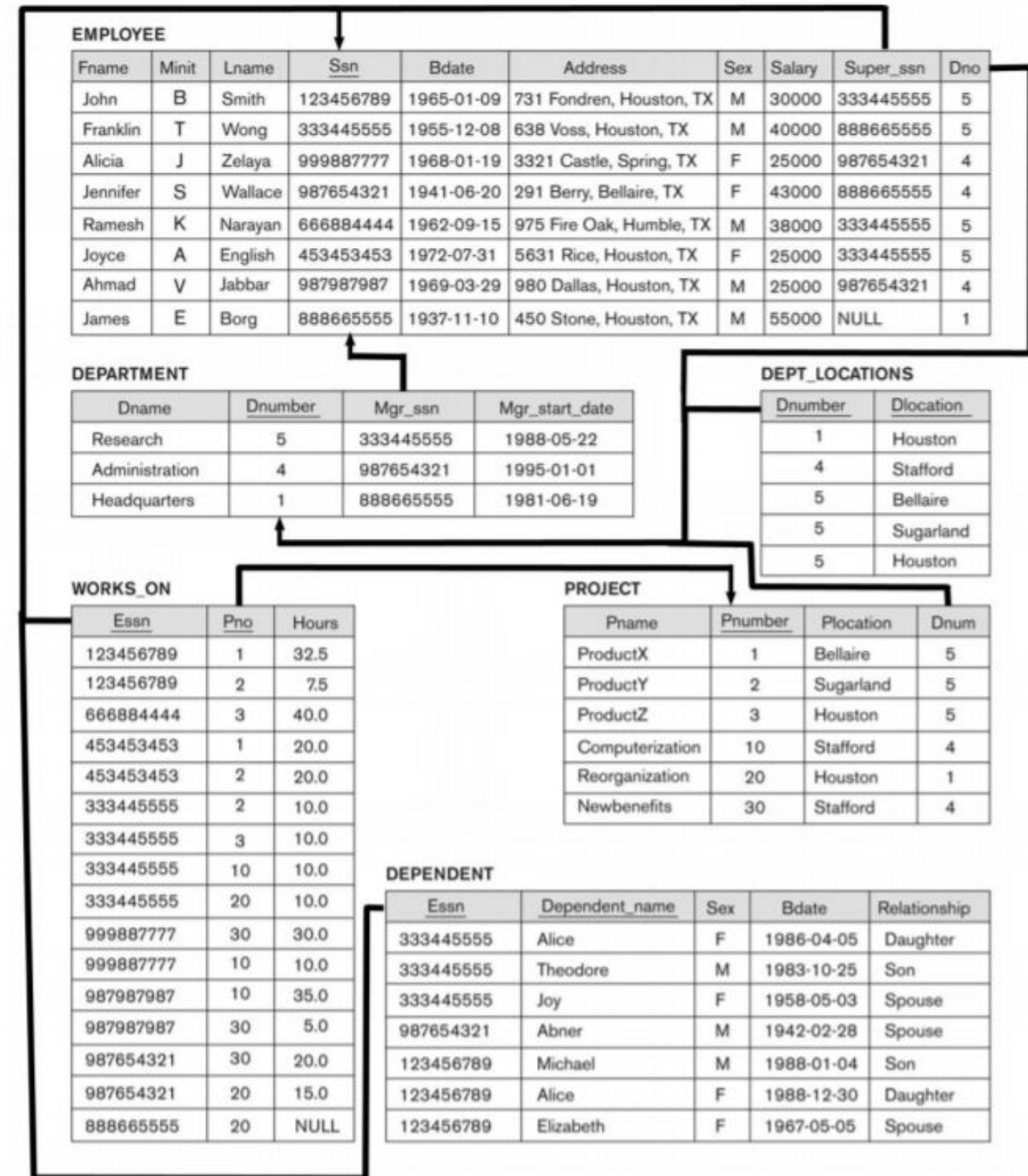


Can we delete the following?

Any tuples in EMPLOYEE with Ssn = '333445555'

Answer:

- No, this deletion will result in even worse Referential Integrity violations
- The tuple involved is referenced by tuples from the EMPLOYEE, DEPARTMENT, WORKS_ON, and DEPENDENT relations



Cascading Deletes

- An option to address Delete operations which violate Referential Integrity is to cascade, or propagate, the deletion
- For instance, in example 2:
 - Delete any tuples in EMPLOYEE with Ssn = '999887777'
- The DBMS could automatically delete the offending tuples from WORKS_ON
 - in addition to the original tuple in EMPLOYEE
 - This must be implemented carefully, as it can lead to unintentional loss of data



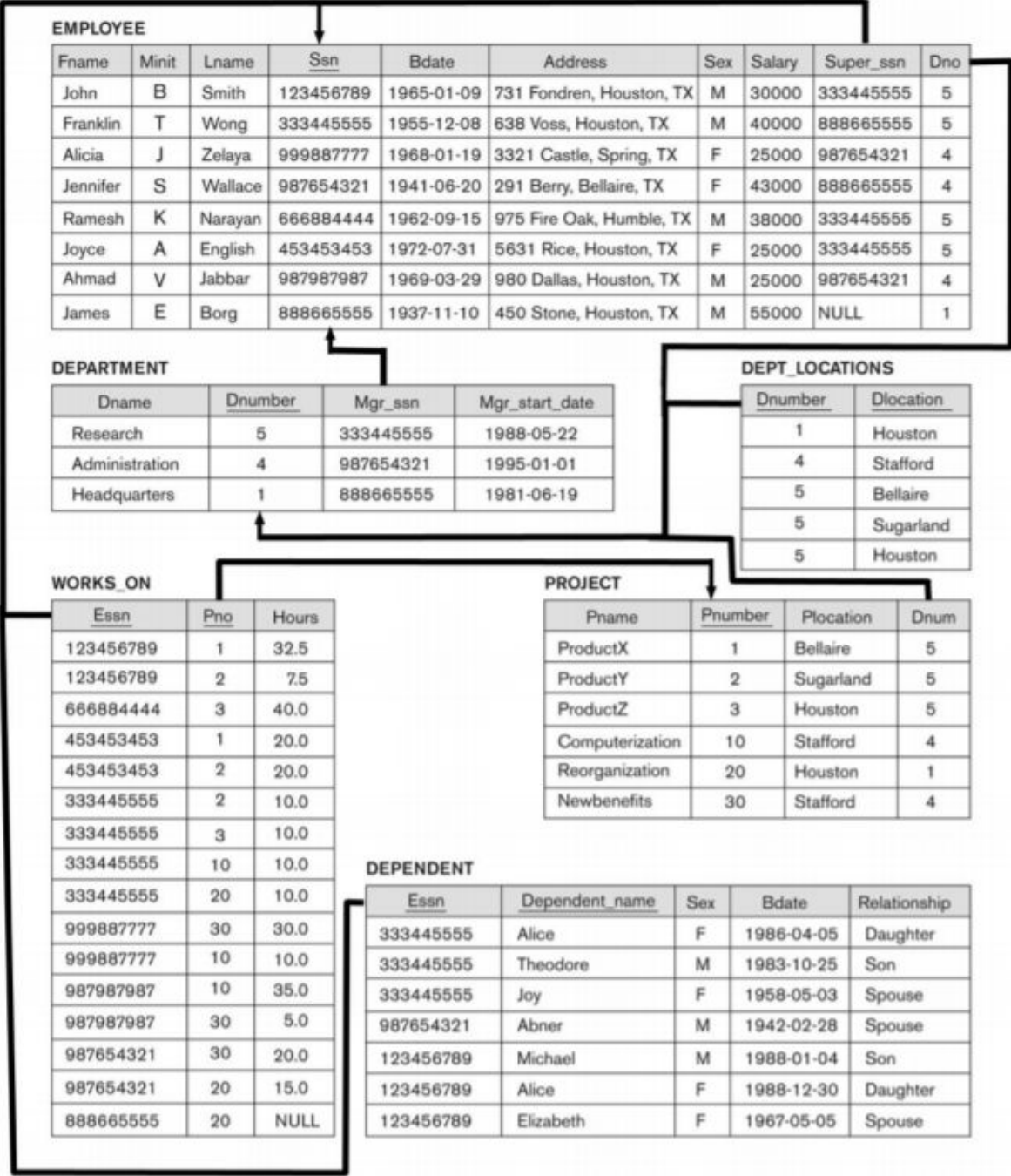
Constraint Violation - Update

- An Update operation is used to change the values of one or more attributes of a relation
- To specify an update, a condition on the attributes of a relation is created which selects one or more tuples to be modified
- Updates can violate all the integrity constraints that we have discussed
 - Key
 - Entity Integrity
 - Referential Integrity



Can we perform the following?

Update the salary of any EMPLOYEE tuples with
Ssn = '999887777' to 28000

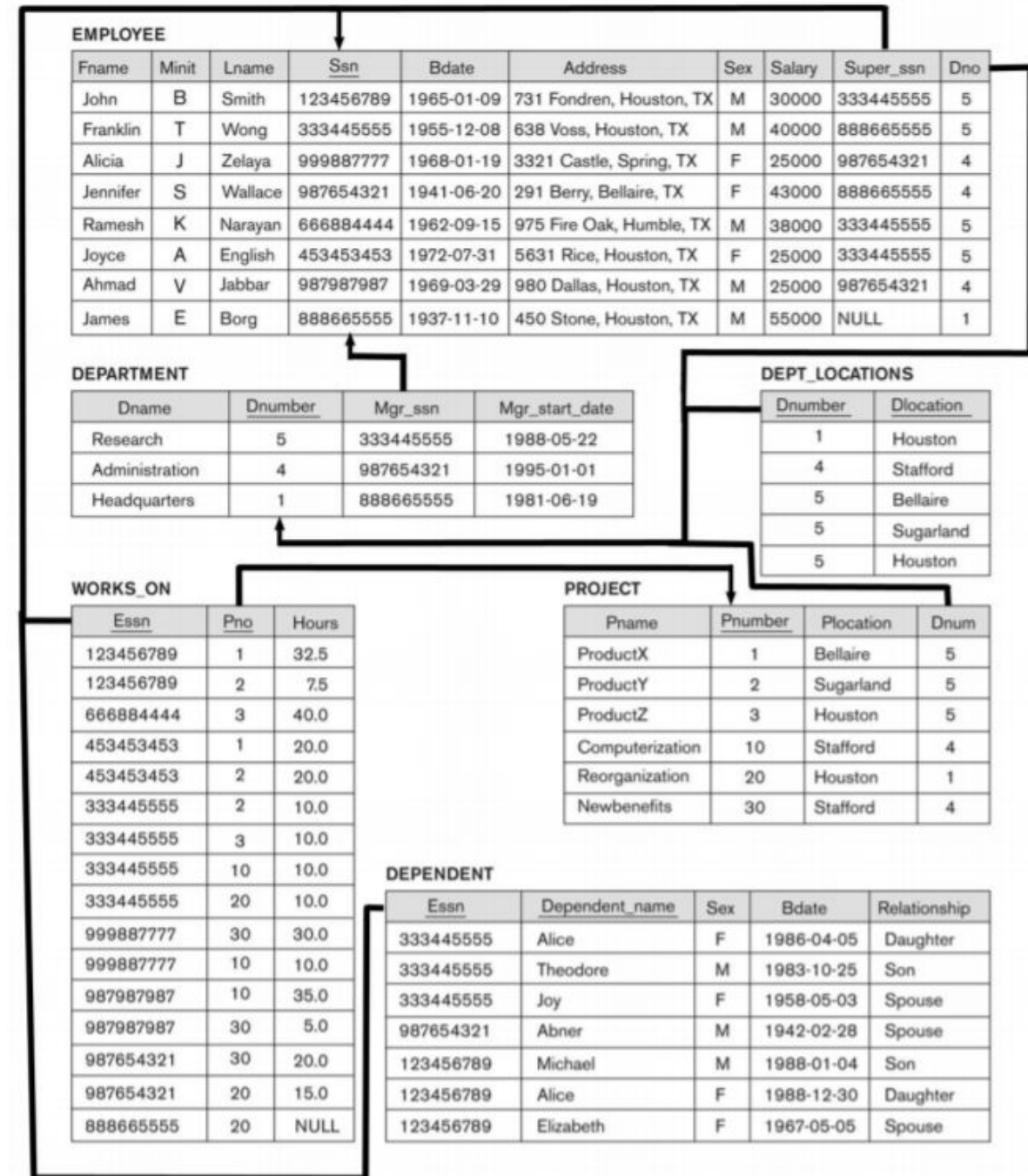


Can we perform the following?

Update the salary of any EMPLOYEE tuples with
Ssn = '999887777' to 28000

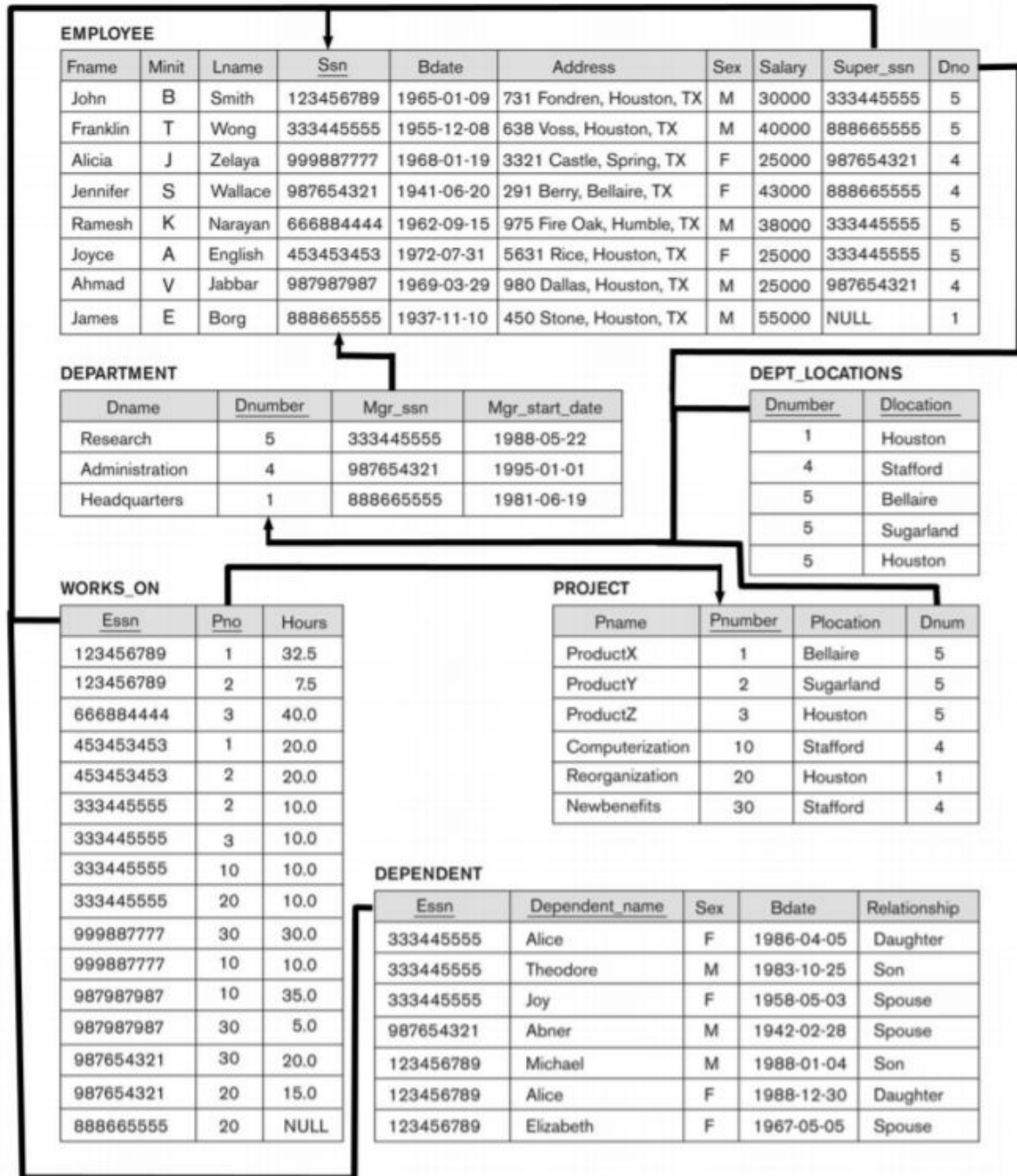
Answer:

- Yes, this modification is acceptable and updates exactly one tuple



Can we perform the following?

Update the Dno of any EMPLOYEE tuples with
Ssn = '999887777' to 1

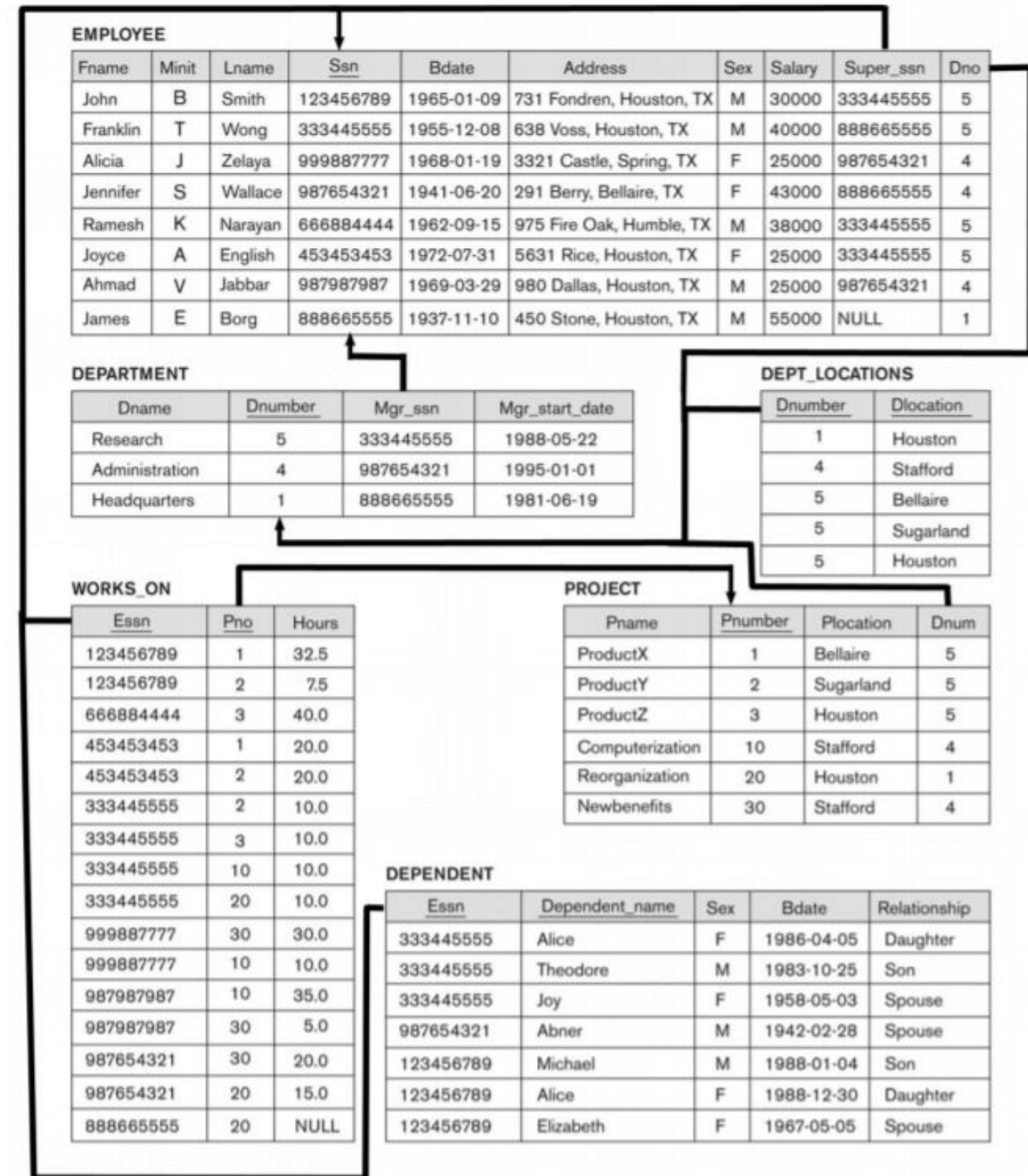


Can we perform the following?

Update the Dno of any EMPLOYEE tuples with
Ssn = '999887777' to 1

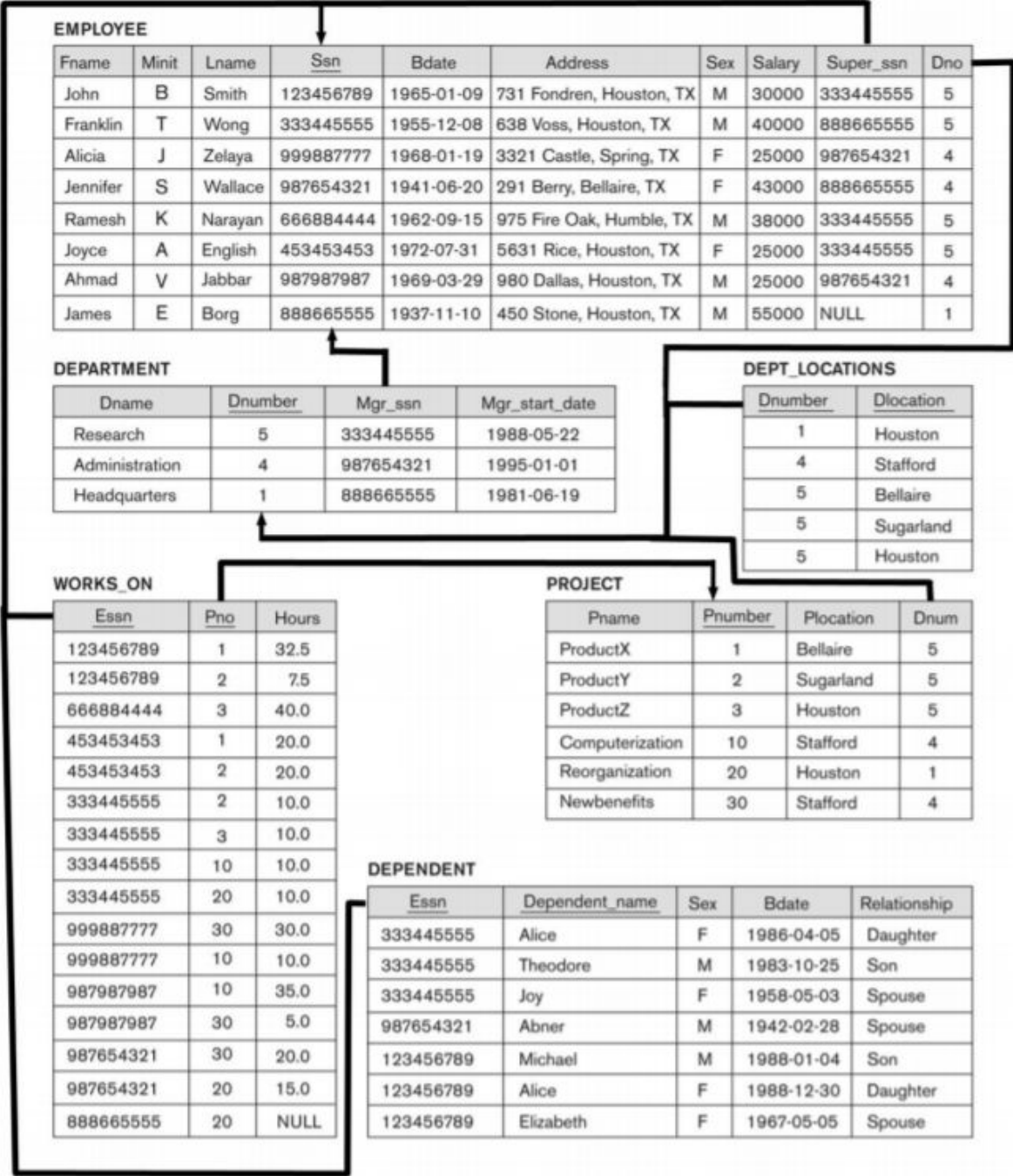
Answer:

- Yes, this modification is acceptable and updates exactly one tuple



Can we perform the following?

Update the Dno of the EMPLOYEE tuple with
Ssn = '999887777' to 7

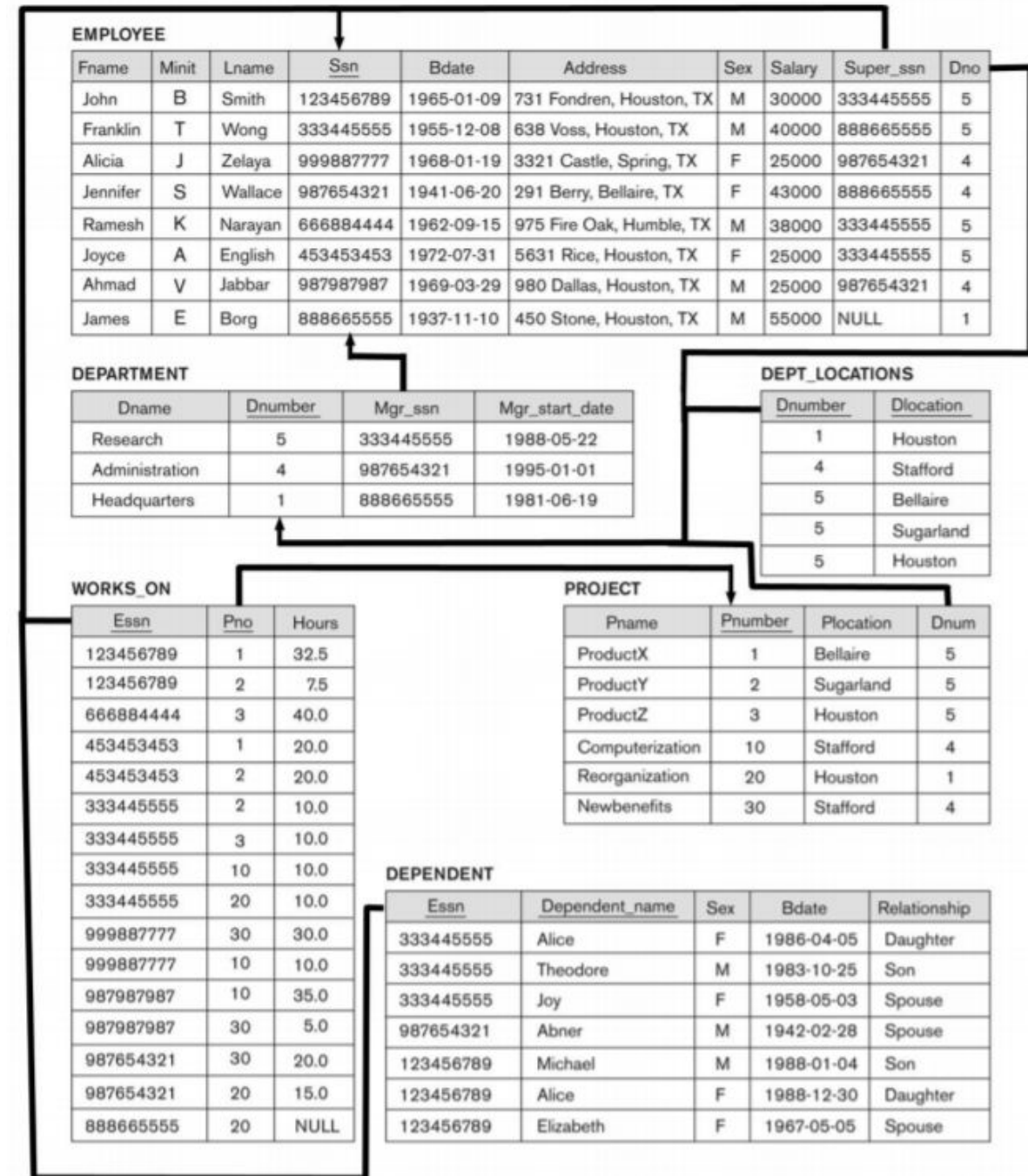


Can we perform the following?

Update the Dno of the EMPLOYEE tuple with
Ssn = '999887777' to 7

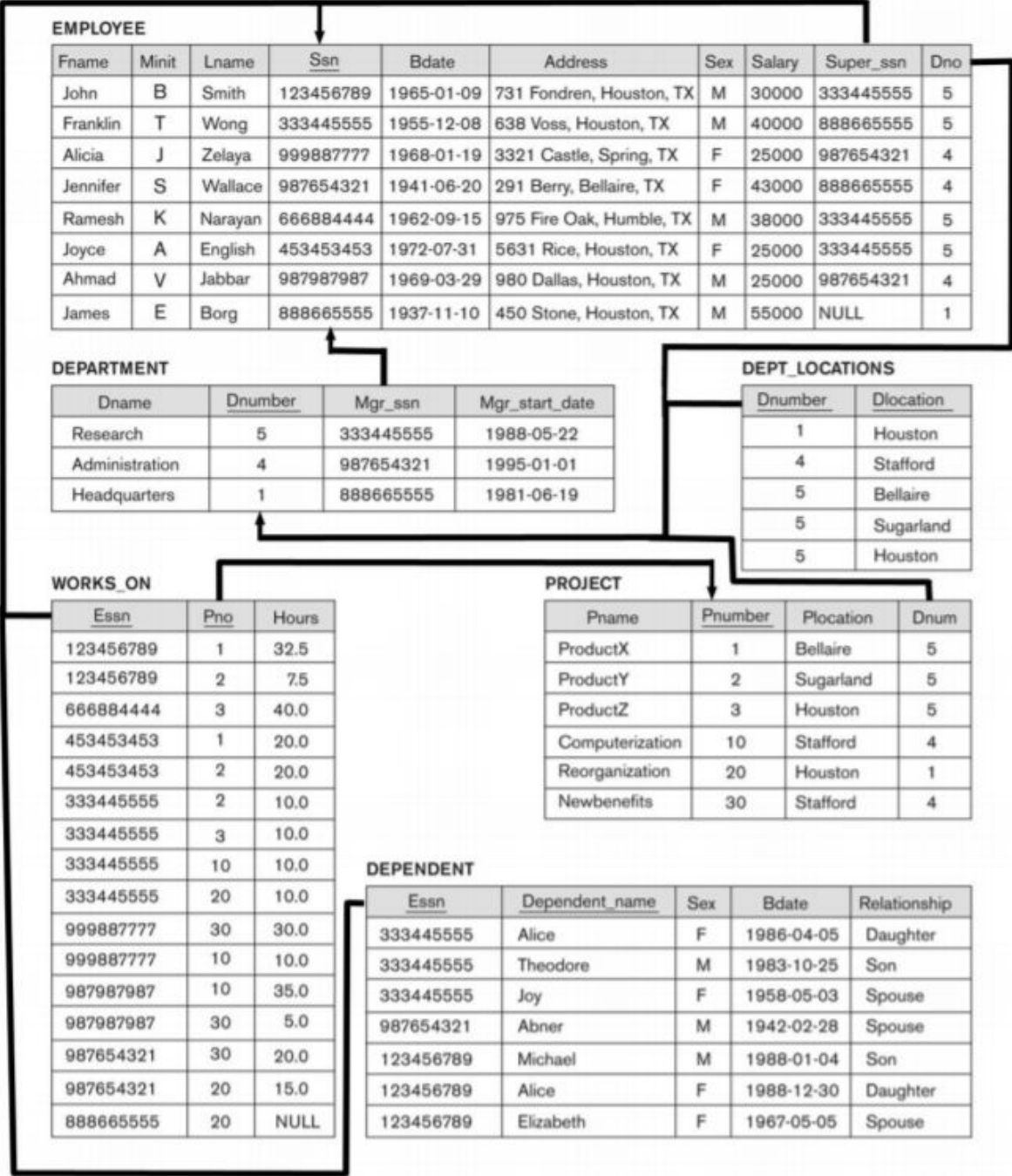
Answer:

- No, this update violates Referential Integrity
- There is no entry in the DEPARTMENT relation with a Dnumber of 7



Can we perform the following?

Update the Ssn of any EMPLOYEE tuples with Ssn = '999887777' to '987654321'

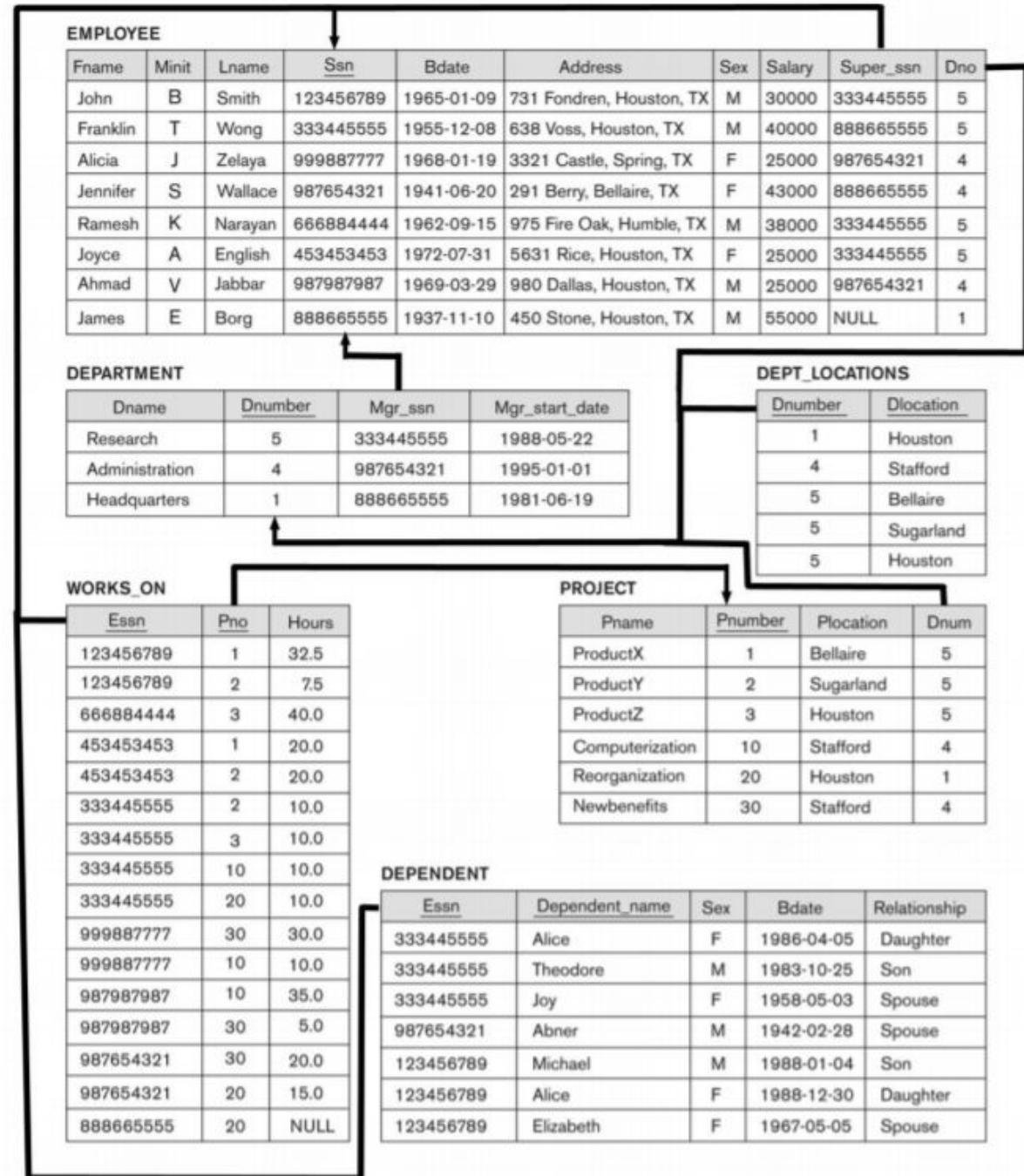


Can we perform the following?

Update the Ssn of any EMPLOYEE tuples with Ssn = '999887777' to '987654321'

Answer:

- No, this update violates the Key constraint
 - It repeats a value that already exists as a Primary Key in another tuple
 - It also violates Referential Integrity constraints
 - There are other relations that refer to the existing value of Ssn



Cascading Updates

As with Delete, an option to address Update operations which violate Referential Integrity is to cascade, or propagate, the update

- For instance: – Update the Ssn of any EMPLOYEE tuples with Ssn = '333445555' to '123123123'
- The DBMS could automatically update the relations which have a Foreign Key to Ssn – WORKS_ON, DEPARTMENT, DEPENDENT and EMPLOYEE itself



Alternatives to Cascading



- The alternatives to the cascading of updates or deletes are:
 - Rejection of the update or delete as long as foreign key references exist
 - Update of the corresponding foreign key to NULL
 - Update of the corresponding foreign key to some default value



Constraints in SQL



- Constraints specified as part of relation, or table, definition are called **table constraints**
- They are specified on each table individually
- They are typically specified during table creation in the **CREATE TABLE** statement
 - can be added later using **ALTER TABLE**
 - *Constraints that affect more than one table are called **Assertions***



Primary Key Constraints



The PRIMARY KEY constraint specifies the attribute(s) that forms the Primary Key

- For a single attribute, the constraint can directly follow the attribute specification
 - **Dnumber INT PRIMARY KEY**
- Composite keys can be specified at the end of the **CREATE TABLE** statement
 - PRIMARY KEY (Dnumber, Dlocation)



UNIQUE



- As we have seen, there is often more than one candidate key in a relation
- Secondary keys can be specified using the UNIQUE constraint
 - For a single attribute, the constraint can directly follow the attribute specification
 - **Engine_num INT UNIQUE**
 - Composite secondary keys can be specified at the end of the **CREATE TABLE** statement
 - **UNIQUE (Licence_Yr, Licence_Mth, Licence_Day)**



NOT NULL



- By default SQL allows NULLs as attribute values
 - a NOT NULL constraint may be specified if NULLs are not permitted for a specific attribute
 - this is always the case for any attribute that forms part of the Primary Key

CREATE TABLE Person

(PPS char(8) NOT NULL PRIMARY KEY,

Fname varchar(255) NOT NULL,

Lname varchar(255),

Phone int);



More Complex Constraints

- Can be specified using:
 - CHECK
 - ASSERTION
 - TRIGGER



CHECK



- More complex constraints can be specified using the CHECK clause
 - used to restrict the values that can be entered for an attribute
- Each CHECK is specified on one or more attributes from a single table
- The CHECK is performed for every tuple that is inserted or modified



CHECK Clause

- CHECK clauses are specified within the CREATE TABLE statement
- They can be specified on an individual attribute

Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21)

- or on multiple attributes from the same table CHECK
(Dept_create_date <= Mgr_start_date)



Referential Integrity

- Referential Integrity is specified using the FOREIGN KEY clause
 - specified at the end of the CREATE TABLE statement
FOREIGN KEY (Dno)
REFERENCES DEPARTMENT(Dnumber)
 - can also have composite Foreign Keys
FOREIGN KEY(artist, album)
REFERENCES ALBUM(artist, name)



Referential Integrity Violation

- As discussed earlier, Referential Integrity can be violated on update, insert or delete
 - Default action in SQL is to reject the operation
- it is possible to specify an alternate action by attaching a clause to each Foreign Key
 - SET NULL
 - CASCADE
 - SET DEFAULT



Referential Integrity Violation

- Each action must be qualified with either
 - ON DELETE
 - ON UPDATE

```
CREATE TABLE EMPLOYEE (  
  Fname VARCHAR(15) NOT NULL,  
  Lname VARCHAR(15) NOT NULL,  
  Ssn CHAR(9) NOT NULL PRIMARY KEY,  
  Super_ssn CHAR(9),  
  Dno INT NOT NULL DEFAULT 1,  
  FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)  
  ON DELETE SET NULL ON UPDATE CASCADE,  
  FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)  
  ON DELETE SET DEFAULT ON UPDATE CASCADE);
```



Naming Constraints

- Constraints can be named using CONSTRAINT
 - Names must be unique within the schema

```
CREATE TABLE movie
(movie_id INT NOT NULL PRIMARY KEY,
title varchar(255) NOT NULL,
genre CHAR(10),
CONSTRAINT check_movie_type CHECK
(genre IN ('Horror', 'Action', 'Other')));
```



Assertions

- An Assertion is a stand-alone constraint in a schema
 - used to specify a restriction that affects more than one table
- Table constraints (CHECK) are only evaluated if, and only if, the table to which it is attached has some data
 - Assertions are required to be true regardless of whether a table is empty or not



Assertion Syntax

- The general form of the ASSERTION command is:
 CREATE ASSERTION <assertion-name>
 CHECK (<search-condition>)
- ASSERTIONs:
 - are associated with the relations in question
 - are evaluated before an operation can be performed on those relations
 - are violated if false and the operation is not allowed
 - define valid states of a DB
 - are actually stored as rows in the ASSERTIONS table which is part of the system catalog



Evaluation of Assertions

- Assertions are checked at the end of each SQL statement
 - a transaction can be more than one SQL statement
 - Assertion evaluation can be deferred until the end of a transaction, but is always evaluated prior to the completion of a transaction
- If an assertion fails, the DBMS returns an error message and the SQL statement is rejected

Assertions

MOVIE

<u>ID</u>	Title	Director	Genre	Cost_Price	Sale_Price	...
-----------	-------	----------	-------	------------	------------	-----

MUSIC

<u>ID</u>	Title	Artist	Genre	Cost_Price	Sale_Price	...
-----------	-------	--------	-------	------------	------------	-----

```
CREATE ASSERTION maximum_inventory  
CHECK((SELECT SUM(Cost_Price) from MOVIE) +  
      (SELECT SUM(Cost_Price) from MUSIC)  
      < 500000);
```


Triggers

- Triggers are Event-Condition-Action rules
 - allow constraints to be checked on specified events and resulting actions to be invoked
- Triggers are only tested when certain events occur
 - e.g. insert, update etc.
- When triggered, a specified condition is tested
 - If the condition does not hold, then no further action is taken in response to the event
 - If the condition is satisfied, defined actions associated with the trigger are performed by the DBMS

Triggers

- General form:

```
CREATE TRIGGER <trigger name>  
( AFTER | BEFORE ) <triggering events> ON  
<table name> [ FOR EACH ROW ]  
[ WHEN <condition> ]  
<trigger actions> ;
```

- <trigger event>

```
INSERT | DELETE | UPDATE [ OF <column name>  
{ , <column name> } ]
```



Triggers

```
CREATE TRIGGER Total_Salary  
AFTER DELETE ON EMPLOYEE  
FOR EACH ROW  
WHEN (:OLD.Dno IS NOT NULL)  
    UPDATE DEPARTMENT  
    SET Total_salary = Total_salary – :OLD.Salary  
    WHERE Dno = :OLD.Dno;
```



Triggers

```
CREATE TRIGGER Inform_Accounts  
BEFORE INSERT OR UPDATE OF Salary, Supervisor_ssn  
    ON EMPLOYEE  
FOR EACH ROW  
WHEN (:NEW.Salary > (SELECT Salary FROM EMPLOYEE  
    WHERE Ssn = :NEW.Supervisor_ssn))  
    salary_violation(:NEW.Supervisor_ssn, :NEW.Ssn);
```



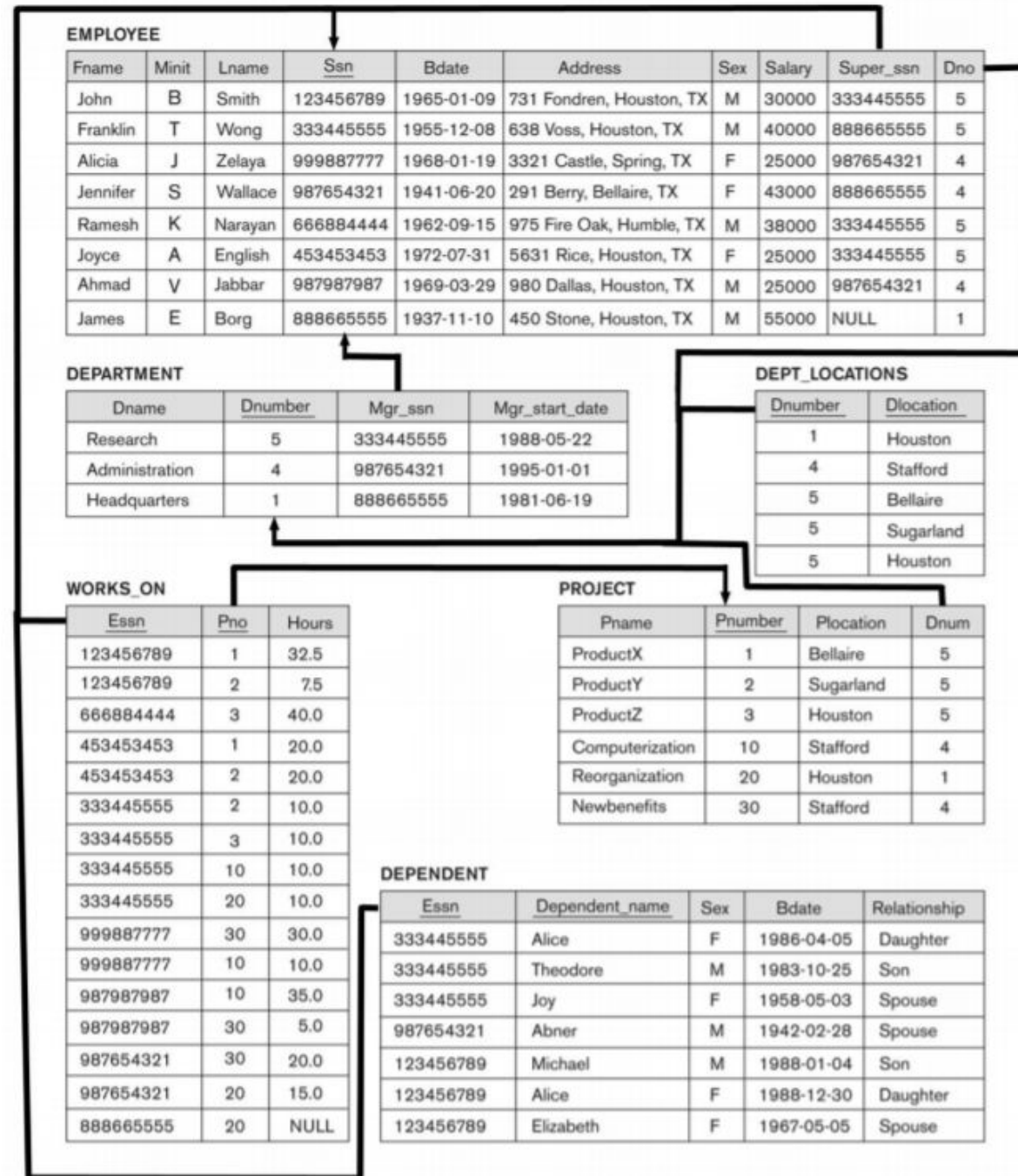
Assertions vs Triggers

- Assertions
 - do not modify the data, only check certain conditions
- Triggers
 - are more powerful because they can check conditions and also modify the data
 - are linked to specific tables and specific events
- All assertions can be implemented as triggers
- Not all triggers can be implemented as assertions
- Oracle does not have assertions



Task 5

- A) Insert into Project relation:
< 'Awesome Project', NULL, 'London', 5 >
- B) Insert into Works_on relation:
<'999887777',20,NULL>
- C) Insert into Dependent relation:
<'123456789','Elizabeth','F','1991-05-15','Daughter'>
- D) Insert into Employee relation:
<'Cecilia', 'F', 'Kolonsky', '677678989', '1960-04-05', '6357 Windswept, Katy, TX', F, 28000, '987654321', 8>
- E) Any Dependent with ESSN = '123456789'
- F) Any tuples in Department with Dno = 1
- G) Any tuples in Employee with SSN = '123456789'
- H) Set the Mgr_ssn in Department relation to NULL for any Department with Dno = 5
- I) Update any Dependent with ESSN='333445555' and Name = 'Theodore' to '1983-09-25'
- J) Update the Pno to 25 of all rows with ESSN='123456789' in the Works_on relation



NAME:

ID:

Please write this in BLOCK CAPS at the top of the page & use portrait

Name:

ID:

- A) YES/NO, reason why
- B) ...



CSU34041 / CSU44D01

Database Constraints

Yvette Graham
ygraham@tcd.ie



Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin