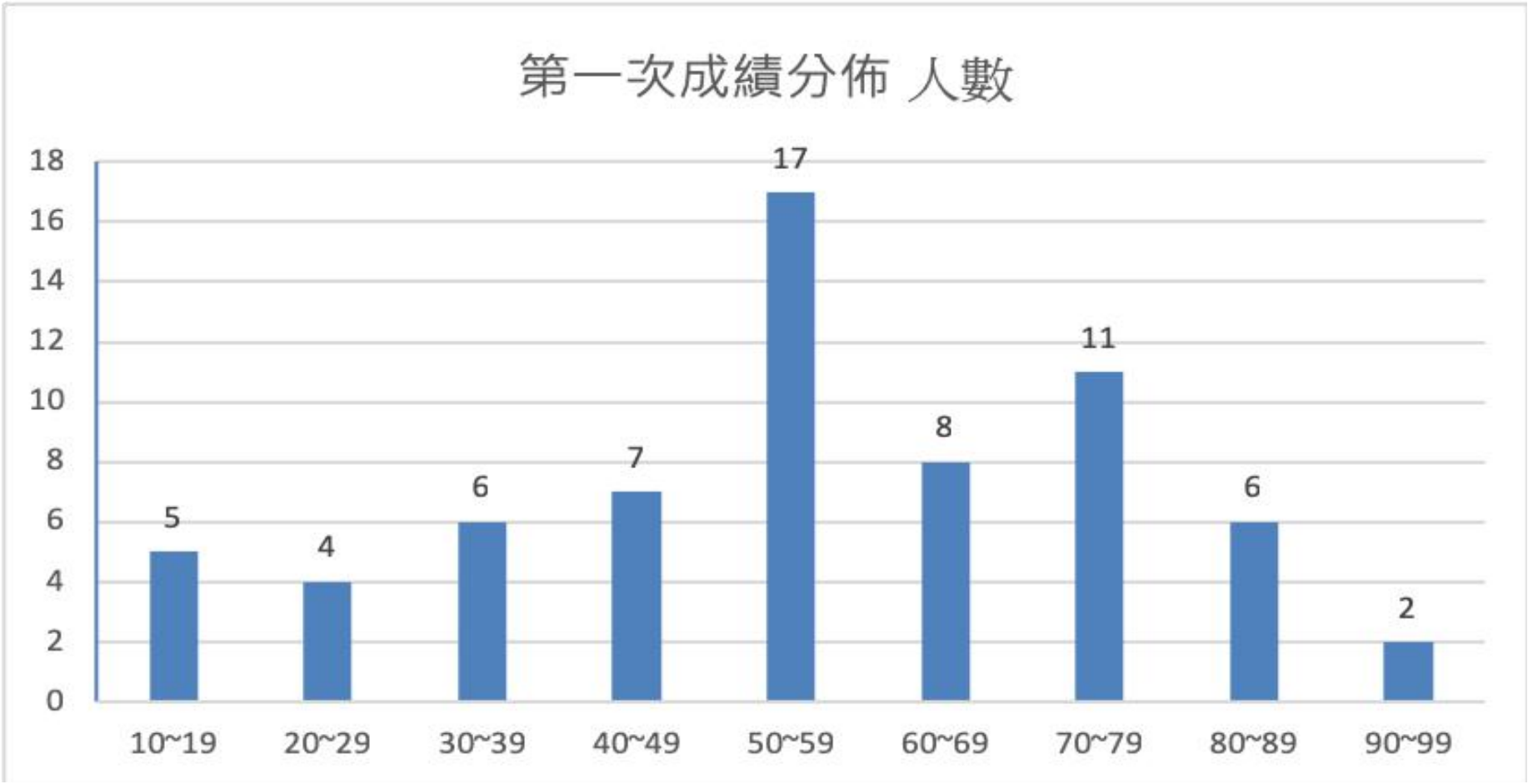

第一次成績分佈 人數
18
16
14
12
10
8
6
4
2
0
5
4
6
7
17
8
11
6
2
10~19
20~29
30~39
40~49
50~59
60~69
70~79
80~89
90~99

Max: 97　Avg: 55

c

1. Which keyword can be used to check whether the result of a correlated nested query is empty or not? (5%)
   (a) DISTINCT (b) ALL (c) EXISTS (d) IN.

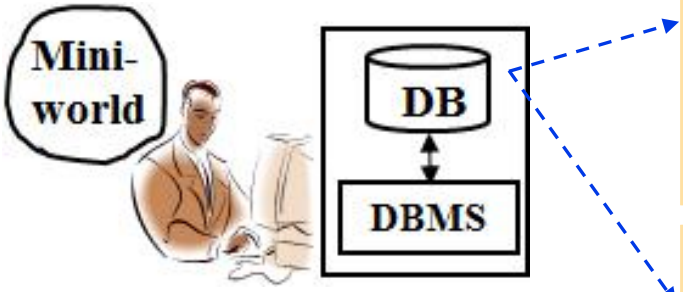
a d

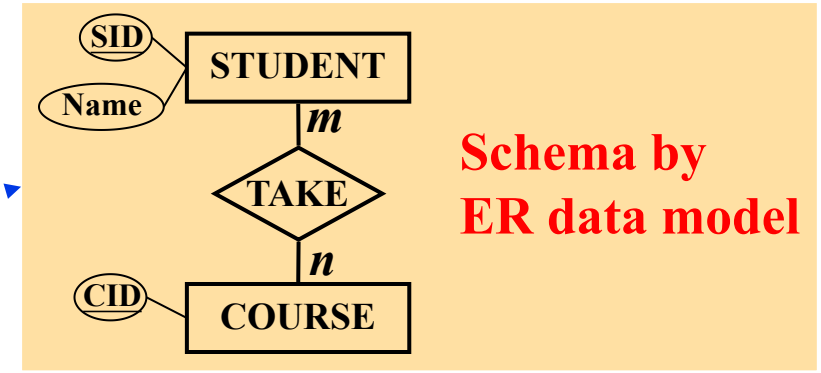
2. Which mechanisms can be used to enforce business rules? (5%)
   (a) CREATE ASSERTION (b) ALTER TABLE (c) CREATE TABLE (d) CREATE TRIGGER.

3. Define the following terms: (a) data model (b) superkey (c) naïve or parametric end-users (d) internal schema. (20%)
4. In case of integrity violation in update operations on relations, please describe what actions can be taken? (10%)

Data Models

- **Data Model**
  - A set of concepts to describe the *structure* of a database, and certain *constraints* that the database should obey.
    - ER data model: {entity, relationship, attribute, key, …}
    - Relational data model: {relation, tuple, attribute, primary key, …}





Schema by Relational data model

STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

Key Constraints

- **Superkey** of R:
  - **A set of attributes SK** of R such that **no two tuples *in any valid relation instance r(R)* will have the same value for SK.**
  - For any distinct tuples t1 and t2 in r(R), **t1[SK] ≠ t2[SK].**

    SK={LicenseNumber}?, SK={Make}?, SK={LicenseNumber, Make}?
- **Key** of R:
  - A "**minimal**" superkey;
  - A superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.
    - Key1 = {LicenseNumber}, Key2 = {EngineSerialNumber}

      are two keys of relation CAR; Key = {Dnumber, Dlocation}
    - SK1={EngineSerialNumber, Make}, SK2={LicenseNumber, Make, Year}

      are superkeys but ***not*** keys

t1 →
t2 →

CAR	LicenseNumber	EngineSerialNumber	Make	Model	Year
	Texas ABC-739	A69352	Ford	Mustang	96
	Florida TVP-347	B43696	Oldsmobile	Cutlass	99
	New York MPO-22	X83554	Oldsmobile	Delta	95
	California 432-TFY	C43742	Mercedes	190-D	93
	California RSK-629	Y82935	Toyota	Camry	98
	Texas RSK-629	U028365	Jaguar	XJS	98

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Categories of End-users-1

- **Naïve or Parametric**
  - They use previously well-defined functions in the form of "canned transactions" against the database.
  - Examples are ***bank-tellers*** or ***reservation clerks*** who do this activity for an entire shift of operations.

Three-Schema Architecture

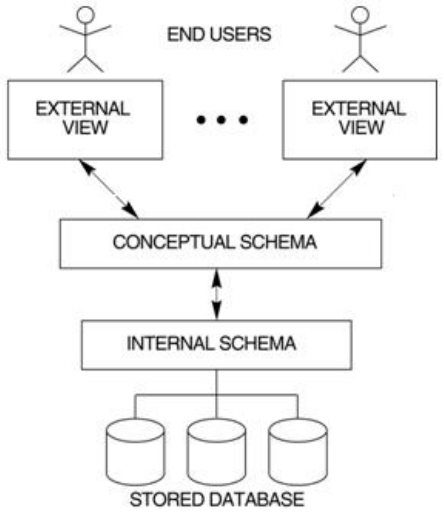
- Defines DBMS schemas at *three levels*:
  - ✓ **External schemas**

    at the external level to describe the various user views. Usually uses the same data model as the conceptual level.
  - ✓ **Conceptual schema**

    at the conceptual level to describe the structure and constraints for the *whole* database for a community of users. Uses a *conceptual* or an *implementation* data model.
  - ✓ **Internal schema**

    at the internal level to describe physical storage structures and access paths. Typically uses a *physical* data model.

DBA defines DB schema



Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (**REJECT** option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (**CASCADE** option, **SET NULL** option, **SET DEFAULT** option)
  - Execute a user-specified error-correction routine

EMPLOYEE (PK) (FK) (FK)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Delete

Modify

DEPARTMENT (PK) (FK)

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0

(FK) (FK)

5. Please describe when not to use a DBMS. (10%)

6. SQL is a non-procedural language and C is a procedural language. Please describe their differences between non-procedural and procedural languages. (10%)

When not to Use a DBMS

- **Main inhibitors (costs) of using a DBMS**:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- **When a DBMS may be unnecessary:**
  - If the database and applications are simple, well defined, and not expected to change.
  - If there are stringent real-time requirements that may not be met because of DBMS overhead.
  - If access to data by multiple users is not required.
- **When no DBMS may suffice:**
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS.

DBMS Languages

- **High Level** or **Non-procedural Languages:**
  - Also called *declarative* languages.
  - e.g., SQL, are *set-oriented* and specify *what* data to retrieve than how to retrieve.
- **Low Level** or **Procedural Languages:**
  - *record-at-a-time*; they specify *how* to retrieve data and include constructs such as looping.

```
SELECT    SNO, NAME
FROM      STUDENT
WHERE     SEX = M
```

```
for (i = 0, TotalStuNum, i++)
    if (Student[i].Sex == M)
        print(Student[i].Sno, Student[i].Name)
```

(a) Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

(b) For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

(c) Retrieve the names of employees who have no dependents.

(d) Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

(e) Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

(f) For each department, retrieve the department number, the number of employees in the department, and their average salary.

(g) Give all employees in department 5 a 10% raise in salary.

(h) Delete the employees whose address is in Douliu, Yunlin.

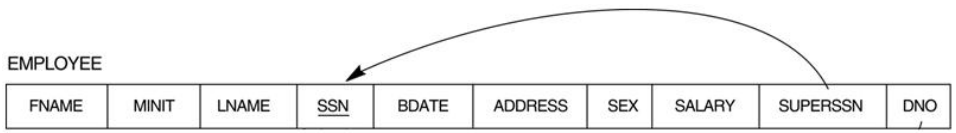
Simple SQL Queries

- Basic SQL queries correspond to using the SELECT, PROJECT, and JOIN operations of the relational algebra
- Example of a simple query on *one* relation
- Query 0: Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.

```
Q0: SELECT   BDATE, ADDRESS
    FROM     EMPLOYEE
    WHERE    FNAME='John' AND MINIT='B' AND LNAME='Smith'
```

  - Similar to a SELECT-PROJECT pair of relational algebra operations; the SELECT-clause specifies the *projection attributes* and the WHERE-clause specifies the *selection condition*
  - However, the result of the query *may contain* duplicate tuples

BDATE	ADDRESS
1965-01-09	731 Fondren, Houston, TX



ALIASES

- Some queries need to refer to the same relation twice
- In this case, ***aliases*** are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

```
Q8:  SELECT  E.FNAME, E.LNAME, S.FNAME, S.LNAME
     FROM    EMPLOYEE  E   S
     WHERE   E.SUPERSSN=S.SSN
```

  - In Q8, the alternate relation names E and S are called *aliases* or *tuple variables* for the EMPLOYEE relation
  - We can think of E and S as two *different copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*
  - Aliasing can also be used in any SQL query for convenience
    Can also use the AS keyword to specify aliases

```
Q8:  SELECT  E.FNAME, E.LNAME, S.FNAME, S.LNAME
     FROM    EMPLOYEE AS E, EMPLOYEE AS S
     WHERE   E.SUPERSSN=S.SSN
```

E

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5

S

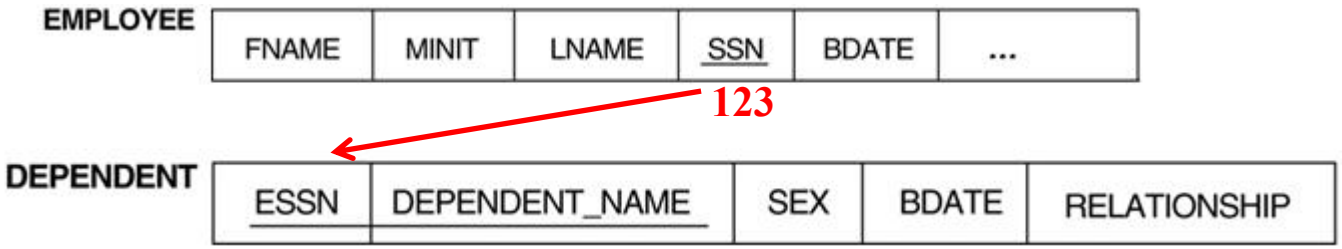
EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5

THE EXISTS FUNCTION

- Query 6: Retrieve the names of employees who have no dependents.

```
Q6: SELECT   FNAME, LNAME
    FROM     EMPLOYEE
    WHERE    NOT EXISTS ( SELECT   *
                          FROM     DEPENDENT
                          WHERE    SSN=ESSN)
```

  - In Q6, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If *none exist* , the EMPLOYEE tuple is selected
  - EXISTS is necessary for the expressive power of SQL



ORDER BY

- The **ORDER BY** clause is used to sort the tuples in a query result based on the values of some attribute(s)

  Query 15: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

```
Q15:  SELECT     DNAME, LNAME, FNAME, PNAME
      FROM       DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
      WHERE      DNUMBER=DNO AND SSN=ESSN AND PNO=PNUMBER
      ORDER BY   DNAME, LNAME
```

- The default order is in ascending order of values
- Keyword **DESC** if we want a descending order;
- Keyword **ASC** can be used to explicitly specify ascending order, even though it is the default

ARITHMETIC OPERATIONS

- The standard arithmetic operators '+', '-'. '*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result
- Query 13: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

```
Q13: SELECT  FNAME, LNAME, 1.1*SALARY
     FROM    EMPLOYEE, WORKS_ON, PROJECT
     WHERE   SSN=ESSN AND PNO=PNUMBER AND
             PNAME='ProductX'
```

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

GROUPING (cont.)

- Query 24: For each department, retrieve the department number, the number of employees in the department, and their average salary.

```
Q24: SELECT     DNO, COUNT (*), AVG (SALARY)
     FROM       EMPLOYEE
     GROUP BY   DNO
```

  - the EMPLOYEE tuples are divided into groups--each group having the same value for the grouping attribute DNO
  - The COUNT and AVG functions are applied to each such group of tuples separately
  - The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples

FNAME	MINIT	LNAME	SSN	• • •	SALARY	SUPERSSN	DNO
John	B	Smith	123456789		30000	333445555	5
Franklin		Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	• • •	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	null	1

Grouping EMPLOYEE tuples by the value of DNO.

DNO	COUNT (*)	AVG (SALARY)
5	4	33250
4	3	31000
1	1	55000

Result of Q24.

UPDATE (cont.)

- Example: Give all employees in department 5 a 10% raise in salary.

```
U6:  UPDATE  EMPLOYEE
     SET     SALARY = SALARY *1.1
     WHERE   DNO  = 5
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple
- The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
- The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DELETE (cont.)

U4C: **DELETE FROM** EMPLOYEE
WHERE ADDRESS **LIKE** '%Douliu, Yunlin%'

A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1