TOPIC: THE RELATIONAL MODEL

CT230
Database
Systems

Recall …
why learn about relational DBMS?

90% of industry/enterprise/business applications are STILL Relational DBMS or Relational DBMS with extensions (e.g. OO Relational).

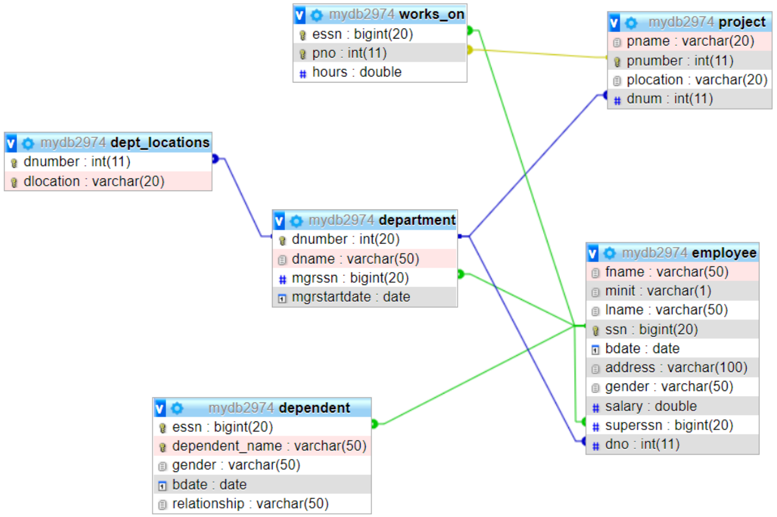
Majority of industry applications require:

- Correctness
- Completeness
- Efficiency (Complex optimisation techniques and complex Indexing structures).

Relational DBMS provide this.

OUR NOTATION

case **not** significant;
spaces not allowed



mydb2974 works_on
essn : bigint(20)
pno : int(11)
hours : double

mydb2974 project
pname : varchar(20)
pnumber : int(11)
plocation : varchar(20)
dnum : int(11)

mydb2974 dept_locations
dnumber : int(11)
dlocation : varchar(20)

mydb2974 department
dnumber : int(20)
dname : varchar(50)
mgrssn : bigint(20)
mgrstartdate : date

mydb2974 employee
fname : varchar(50)
minit : varchar(1)
lname : varchar(50)
ssn : bigint(20)
bdate : date
address : varchar(100)
gender : varchar(50)
salary : double
superssn : bigint(20)
dno : int(11)

mydb2974 dependent
essn : bigint(20)
dependent_name : varchar(50)
gender : varchar(50)
bdate : date
relationship : varchar(50)

DATABASE SCHEMA

employee(fname, minit, lname, ssn, bdate, address, gender, salary, superssn, dno)

department(dname, dnumber, mgrssn, mgrstartdate)

dept_locations(dnumber, dlocation)

project(pname, pnumber, plocation, dnum)

works_on(essn, pno, hours)

dependent(essn, dependent_name, gender, bdate, relationship)

SETTING UP YOUR DATABASE …

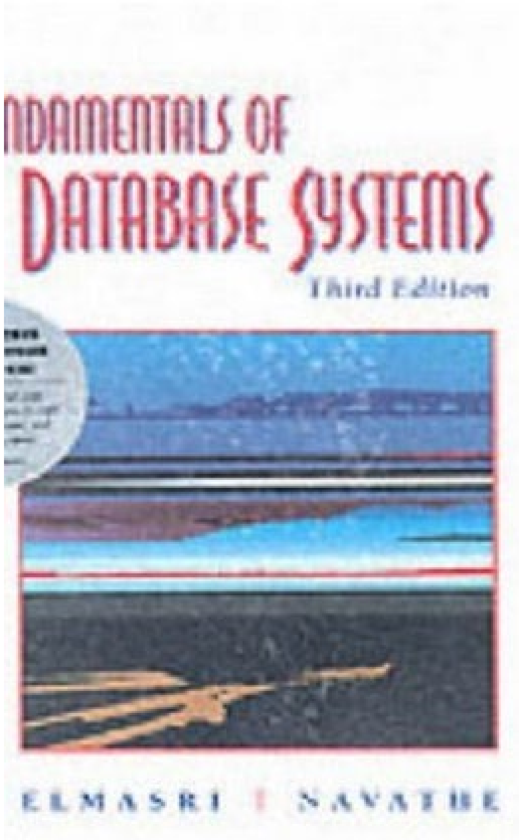
See supplemental notes and video will be added before labs next week ….

TOPIC:
Defining and working with the Relational Model

See

Elmasri and Navathe book

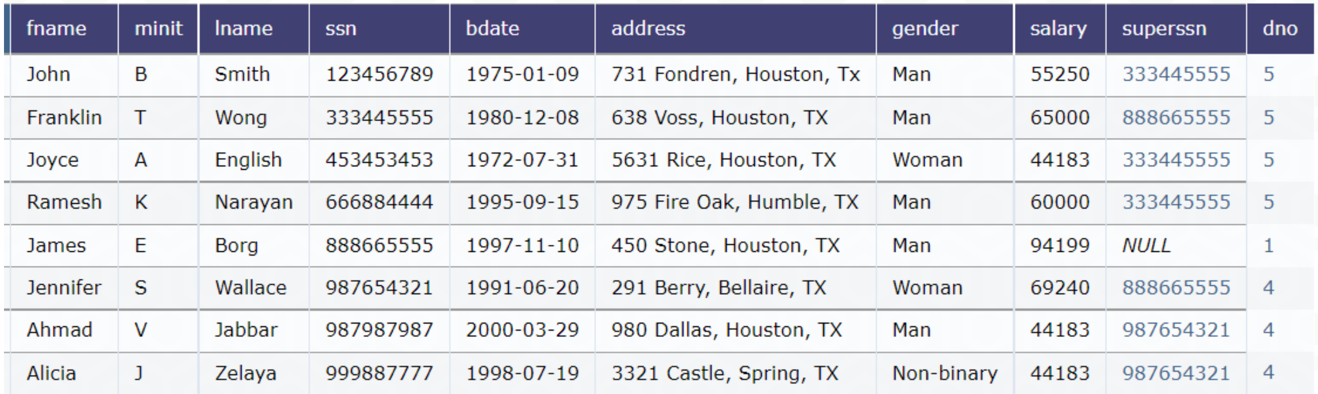
Chapter 7



RELATIONAL DATA MODEL

- Collection of relations (often called *tables*) where each relation contains tuples (rows) and attributes (columns).
- Closely related to file system model at (we use in our own programming)
- Relations are named: e.g., relation ‘employee’:

employee(fname, minit, lname, ssn, bdate, address, gender, salary, superssn, dno)



fname	minit	lname	ssn	bdate	address	gender	salary	superssn	dno
John	B	Smith	123456789	1975-01-09	731 Fondren, Houston, Tx	Man	55250	333445555	5
Franklin	T	Wong	333445555	1980-12-08	638 Voss, Houston, TX	Man	65000	888665555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	Woman	44183	333445555	5
Ramesh	K	Narayan	666884444	1995-09-15	975 Fire Oak, Humble, TX	Man	60000	333445555	5
James	E	Borg	888665555	1997-11-10	450 Stone, Houston, TX	Man	94199	*NULL*	1
Jennifer	S	Wallace	987654321	1991-06-20	291 Berry, Bellaire, TX	Woman	69240	888665555	4
Ahmad	V	Jabbar	987987987	2000-03-29	980 Dallas, Houston, TX	Man	44183	987654321	4
Alicia	J	Zelaya	999887777	1998-07-19	3321 Castle, Spring, TX	Non-binary	44183	987654321	4

- Relation = table

- Attributes = columns and these are (mostly always) fixed (e.g., fname, minit, lname …) and do not change

* The number of attributes of a relation is referred to as its grade or degree

- Tuples = rows which contain the data and there is variable number of these

* The number of tuples of a relation is referred to as its cardinality.

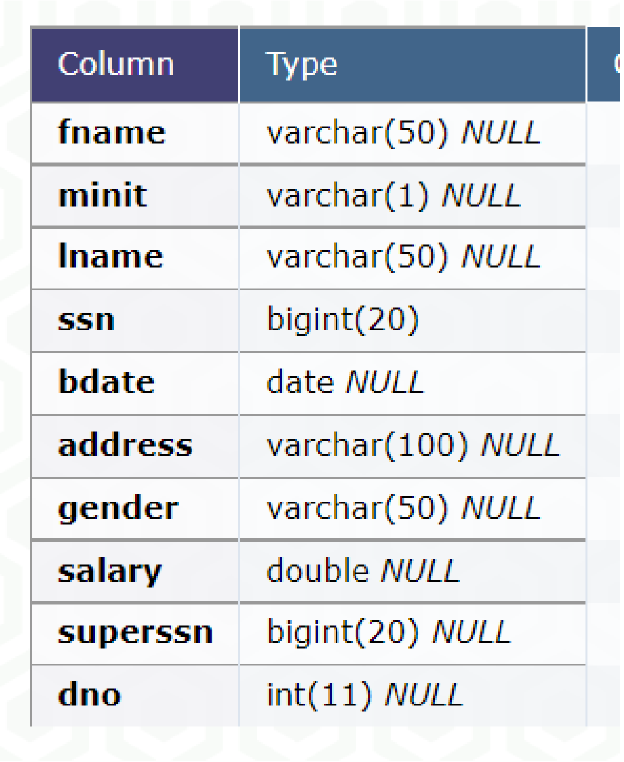
ATTRIBUTES/COLUMNS

Each attribute belongs to **one** *domain* and has a single:

- name
- data type
- format

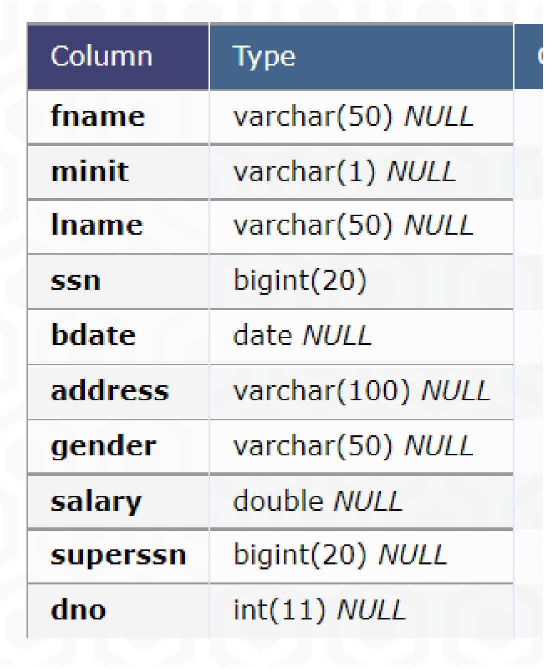
e.g.,

Name:	bDate
Type:	date
Format:	yyyy/mm/dd



Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

NAMING COLUMNS (ATTRIBUTES)



Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

- case **not** significant in SQL
- no spaces allowed
- no reserved keywords (e.g. date) allowed
- as usual, if picking names yourself - choose meaningful variable name
- if given the names of relations and attributes, use **exactly** what you are given

DATA TYPES

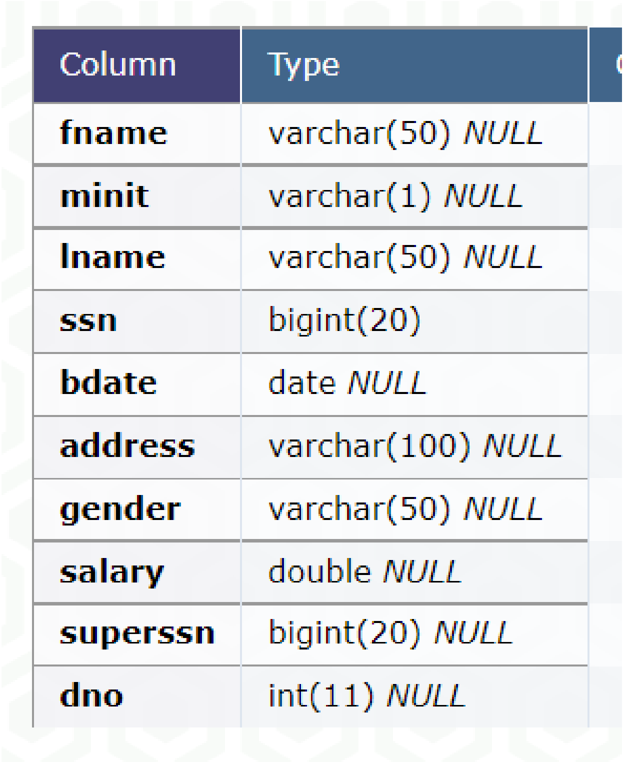
As with many programming languages must specify the data type of all attributes (columns) defined

Common data types used are:

- varchar(*N*), N an integer (for strings)
- date
- int
- double

Often specify the sizes especially for integers and strings

Will discuss in more detail when we start to create tables

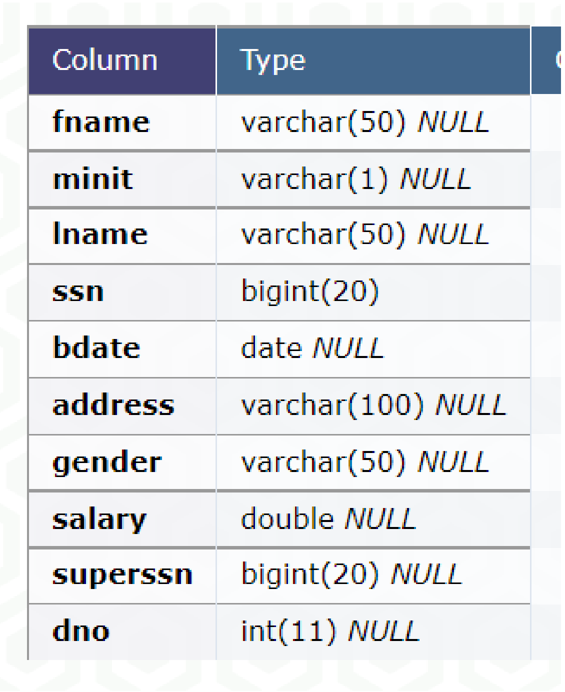


Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

NULL

Null valued-attributes: values of some attribute within a particular tuple may be unknown or may not apply to a particular tuple … null value is used for these cases.

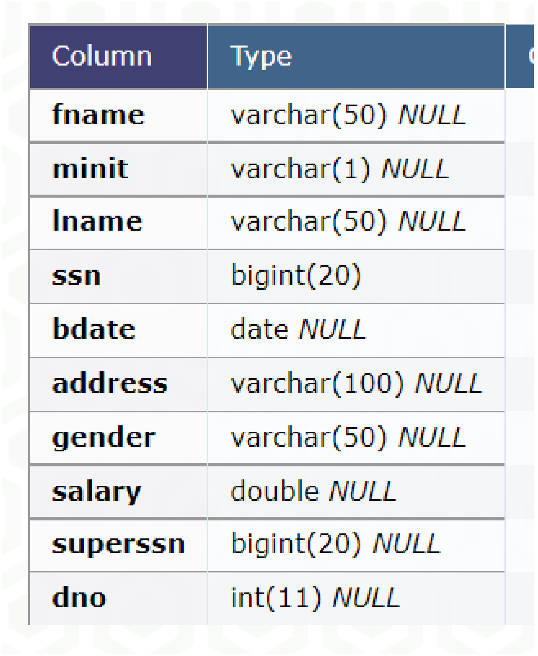
NULL is a special marker used in SQL to denote the absence of a value



Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

- In some cases we wish to allow the possibility of a NULL value although they will often require extra handling (e.g. checking for =NULL).
- In other cases we want to prevent NULL being entered as a value and specify NOT NULL as a constraint on data entry.

ATOMIC ATTRIBUTES



Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

An **atomic attribute** is an attribute which contains a single value of the appropriate type. Generally meaning, “no repeating values of the same type”

The relational model should **only** have atomic values

Example: Attribute address of type varchar(100) *Null*

Should only contain one address “3 Cherry Road, Carlow”

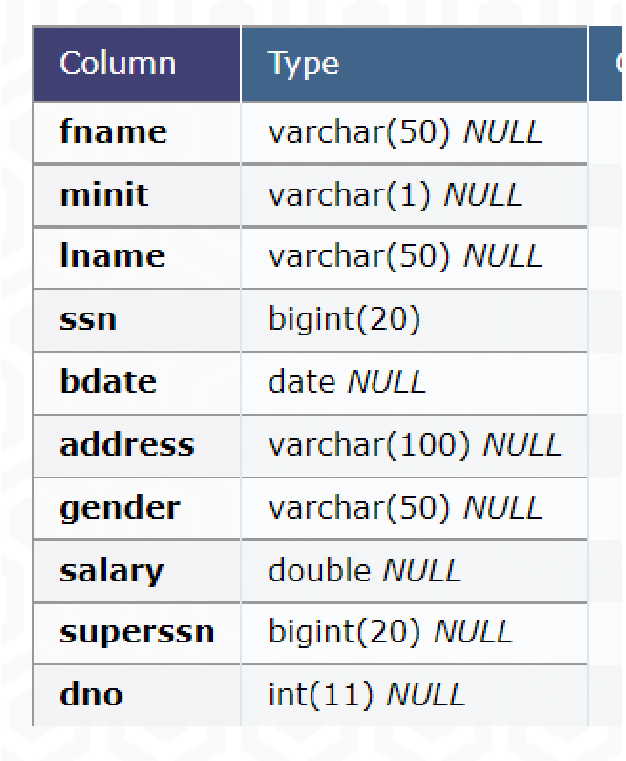
Rather than “3 Cherry Road, Carlow; Apt 12 Corrib Village, Galway”

COMPOSITE ATTRIBUTES

A **composite attribute** is an attribute that is composed of several more basic/atomic attributes.

Example:

- Name = FirstName, Middle Initial, Surname



Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

We often want to decompose a composite attribute into atomic attributes unless there is a very good reason not to (e.g. why is address not decomposed in to street, city, county, etc.?)

MULTI-VALUED ATTRIBUTES

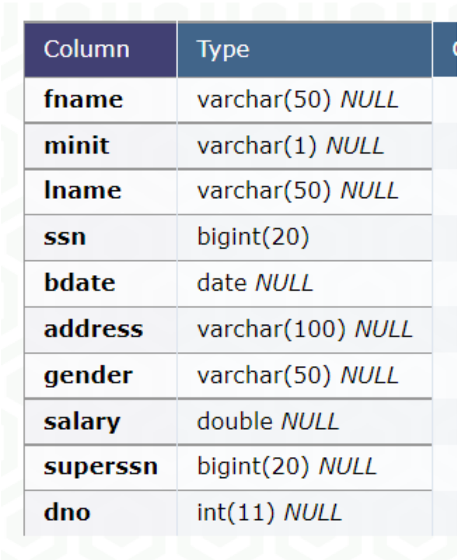
A **multi-valued attribute** is an attribute which has lower and upper bounds on the number of values for an individual entry.

(the opposite of an atomic attribute)

Example:

qualifications

phone numbers



Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

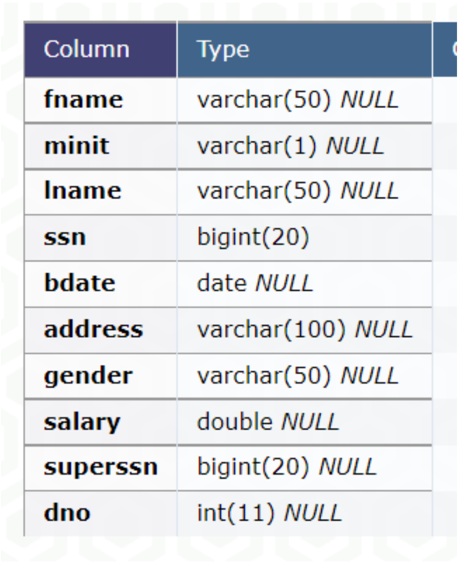
The relational model should **NOT** store multi-valued attributes – database design/re-design should be used to deal with this issue by creating more attributes (columns) or more tables.

DERIVED ATTRIBUTES

A **derived attribute** is an attribute whose value can be determined from another attribute

Example:

from bdate can derive age



Column	Type
fname	varchar(50) *NULL*
minit	varchar(1) *NULL*
lname	varchar(50) *NULL*
ssn	bigint(20)
bdate	date *NULL*
address	varchar(100) *NULL*
gender	varchar(50) *NULL*
salary	double *NULL*
superssn	bigint(20) *NULL*
dno	int(11) *NULL*

It is a good idea to not directly store attributes which can be derived from other attributes.

RECALL ….

- We said that the Relational Data Model consists of a **collection** of relations (*tables*)
- Tables are **cross-linked**

COLLECTION OF RELATIONS

A relational database usually contains many relations (tables) rather than storing all data in one single relation.

A relational database schema, S, is a definition of a set of relations that are to be stored in the database, i.e.,

$S = \{R_1, R_2, \dots, R_n\}$

e.g., S = {employee, department, works_on, dept_locations, project, dependent}

Formal definition of “schema”

A relational schema *R* is the definition of a table in the database. It can be denoted by listing the table name and the attributes:

$R(A_1, A_2, \ldots\ldots, A_n)$

where A_i is an attribute.

e.g. with n=3, that is, 3 attributes:

works_on(essn, pno, hours)

RECALL: Database schemas and instances

Similar to types and variables in programming languages.

Schema: the logical structure of a database.

Instance: the actual content of the database at some point in time

LINKING TABLES …

Two VERY (*very, very*) important concepts within the relational model which allow tables to be linked and cross-referenced are:

- PRIMARY KEY attributes
- FOREIGN KEY attributes

We will define and discuss these tomorrow!

QUESTIONS?/ISSUES?