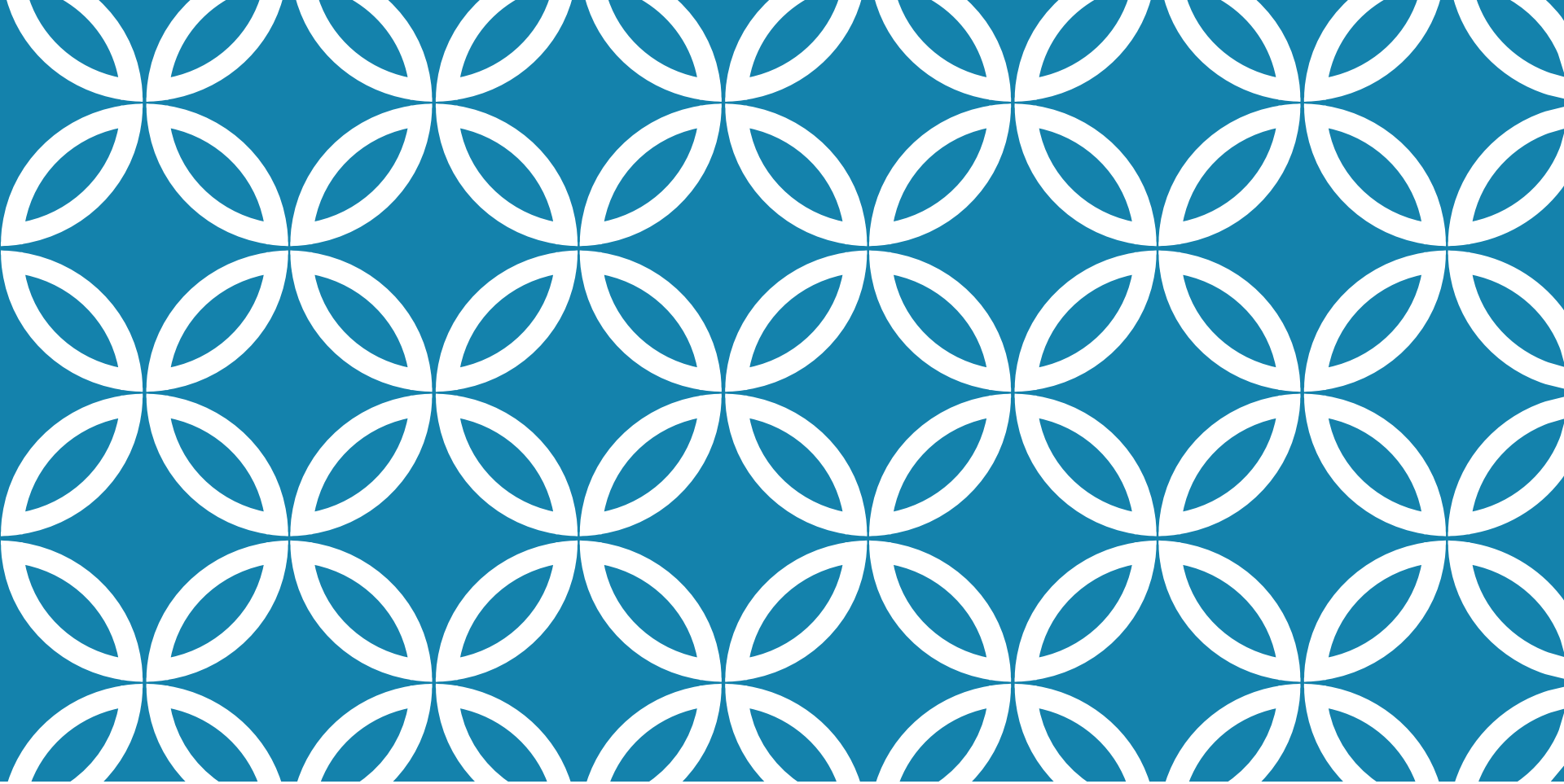


TOPIC: THE RELATIONAL MODEL

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).

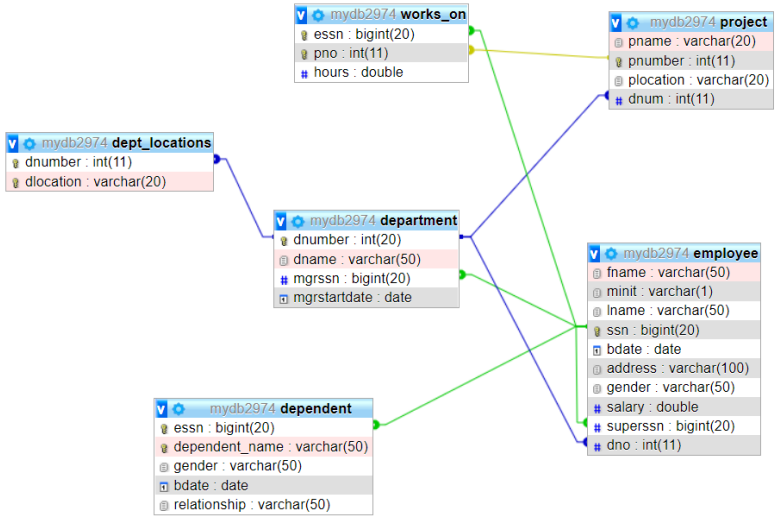
Majority of industry applications require:

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

Relational DBMS provide this.

OUR NOTATION

case **not** significant;
spaces not allowed



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 …

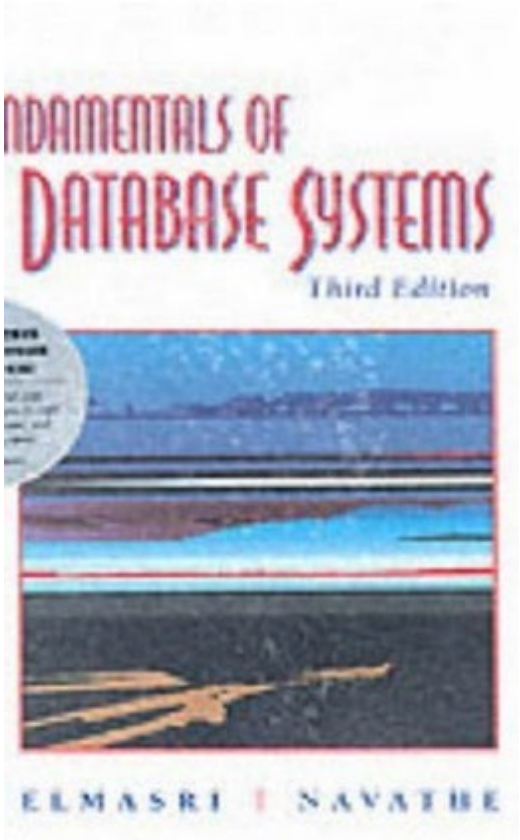
See supplemental notes and video will be added before labs next week ….

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

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 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

- 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

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*

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

Example:

- Name = FirstName, Middle Initial, Surname

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.,

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

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:

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

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 . . .

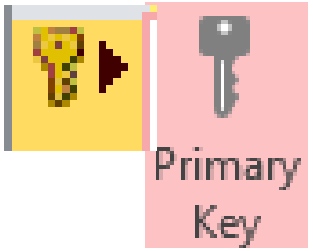
Two VERY (*very, very*) important concepts within the relational model which allow tables to be linked and cross-referenced are:

- PRIMARY KEY attributes
- FOREIGN KEY attributes

We will define and discuss these tomorrow!

QUESTIONS?/ISSUES?

PRIMARY KEYS



Fundamental concept of Primary Keys:

All tuples (row) in a relation must be distinct

To ensure this must have:

- one of more attributes/columns whose data values will always be unique for each tuple - these attributes are called **key attribute(s)** and are used to uniquely identify a tuple in the relation.

There may be a few possibilities for primary key – these are called Candidate keys

One candidate key is ultimately chosen as the primary key as part of the Design stage

DEFINITION: PRIMARY KEY

A primary key is defined as one or more attributes, per table where:

- there can be only one such primary key per table
- the primary key can never contain the NULL value
- all values entered for the primary key must be unique (no duplicates across rows)

- Often primary keys are used as indexes (*will discuss later)
- We use the convention (in writing) that attributes that form the primary key are <u>underlined</u>

EXAMPLES (Company schema): Adminer

Table: employee

Select data | Show structure | Alter table | New item

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

Indexes

PRIMARY	*ssn*

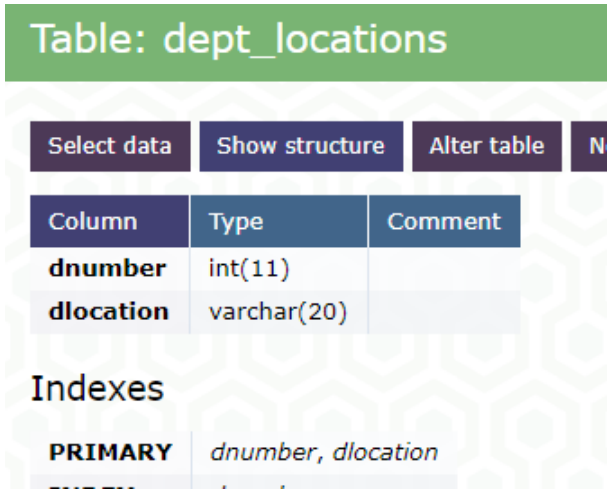


Table: dept_locations

Select data | Show structure | Alter table | N

Column	Type	Comment
dnumber	int(11)	
dlocation	varchar(20)	

Indexes

PRIMARY	*dnumber, dlocation*

MySQL » mysql1.it.nuigalway.ie » mydb2974 » Table:

Table: works_on

Select data | Show structure | Alter table | New item

Column	Type	Comment
essn	bigint(20)	
pno	int(11)	
hours	double *NULL*	

Indexes

PRIMARY	*essn, pno*

What is the primary key of these tables?

See menti.com

Consider the `works_on` table:

A table to hold details on which projects an employee works on and the number of hours worked on each project:

`works_on(essn, pno, hours)`

Primary Key?

" one of more attributes/columns whose data values will always be unique for each tuple."

SOME SAMPLE DATA FROM works_on TABLE ....

essn	pno	hours
123456789	1	32.5
123456789	2	7.5
333445555	2	10

An employee can work on more than one project

A project can contain more than one employee

ALL DATA FROM THE works_on TABLE ....

essn	pno	hours
123456789	1	32.5
123456789	2	7.5
123456789	3	3
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
453453453	1	20
453453453	2	20
666884444	3	40
888665555	20	0
987654321	20	15
987654321	30	20
987987987	10	35
987987987	30	5
999887777	30	30

works_on(essn, pno, hours)

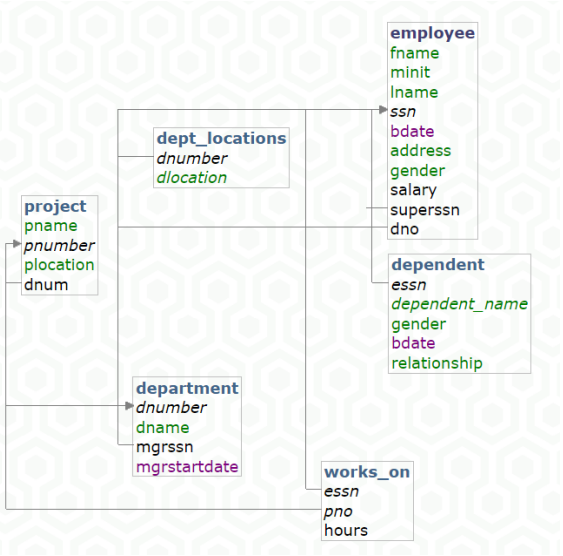
QUESTION: What are suitable primary keys for the following tables?

module(code, name, department, semester, exam_duration, ECTS)

student(ID, FirstName, LastName, HomeAddress, HomePhone)

car(EngineNo, CarReg, Make, Model, Year)

FOREIGN KEYS



Fundamental concept of Foreign Keys:

- Allows data in tables to be linked and cross-referenced by matching the same data values in both tables

Note:

- Matching must take place to primary or candidate keys
- There may be a few different links across the same tables

DEFINITION: FOREIGN KEY

A foreign key is an attribute, or set of attributes, within one table that matches or - links to - the candidate key of some other table (possibly the same table)

More formally - Given relations r_1 and r_2, a foreign key of r_2 is an attribute (or set of attributes) in r_2 where that attribute is a candidate key in r_1. relations r_1 and r_2 may be the same relations

FOREIGN KEY TERMINOLOGY

Often use the terminology of:

- parent, master or referenced table/relation for the relation containing the candidate key(s)
- child or referencing table/relation for the relation containing the foreign key

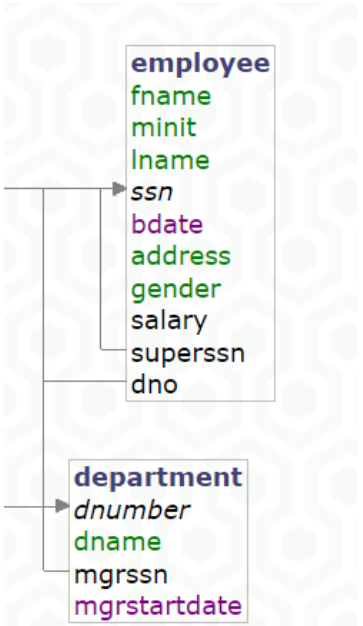
For example:

In company schema, department is parent/master table (containing PK dnumber) and employee is child/referencing table (with FK dno)

Foreign keys

Source	Target	ON DELETE	ON UPDATE	
dno	department(*dnumber*)	RESTRICT	RESTRICT	Alter

EXAMPLE: FOREIGN KEY



employee

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

department

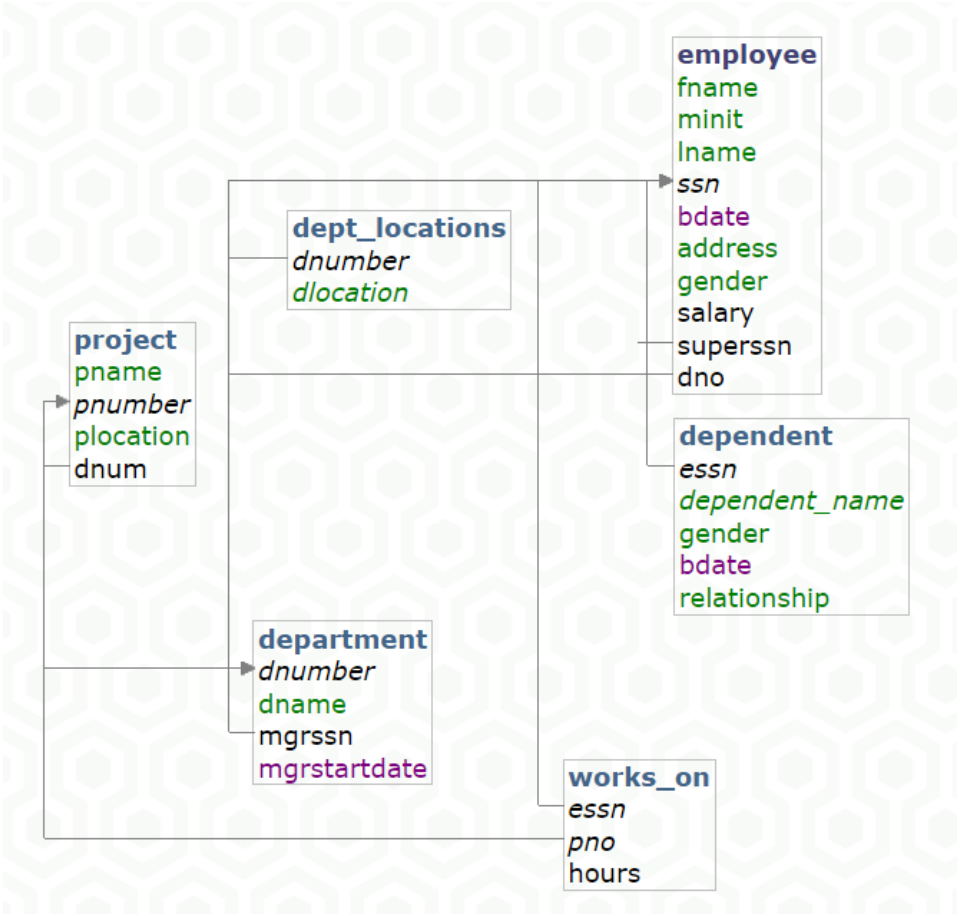
Modify	dnumber	dname	mgrssn	mgrstartdate
edit	1	Headquarters	888665555	2019-06-19
edit	4	Administration	987654321	2015-01-01
edit	5	Research	333445555	2018-05-22

dno is a foreign key in relation employee linking to dnumber in department

EXAMPLES (COMPANY SCHEMA): SEE menti.com

What is/are the foreign key(s) in the dependent table?

What is/are the foreign key(s) in the employee table?



SUMMARY: RELATIONAL MODEL

- Terminology and definitions associated with main concepts of the relational model **very important**
- Company schema will be used **extensively** for much of the course so a good understanding of it from these lectures is **very important**
- VERY important you get access to the CS Intranet and MySQL and import the company database this week if you are registered.
- Next … how to create tables and add data to tables…