

ER-to-relational Mapping

1/28

So far, have considered mappings for ...

- ER attribute → relational attribute
- ER entity → relational table
- ER key → primary key for table
- n:m relationship → relational table
(with foreign key for each participating entity plus relationship attributes)
- 1:n relationship → foreign key plus relationship attributes
- 1:1 relationship → foreign key plus relationship attributes

n-way Relationships

2/28

Relationship mappings above assume binary relationship.

If multiple entities are involved: *You need to have a separate table to the diamond*

- *n:m* generalises naturally to *n:m:p:q* *it does not matter how many ‘manys’ there*
 - include foreign key for each participating entity
 - include any other attributes of the relationship
- other multiplicities (e.g. *1:n:m*) ...
 - need to be mapped the same as *n:m:p:q*
 - so not quite an accurate mapping of the ER

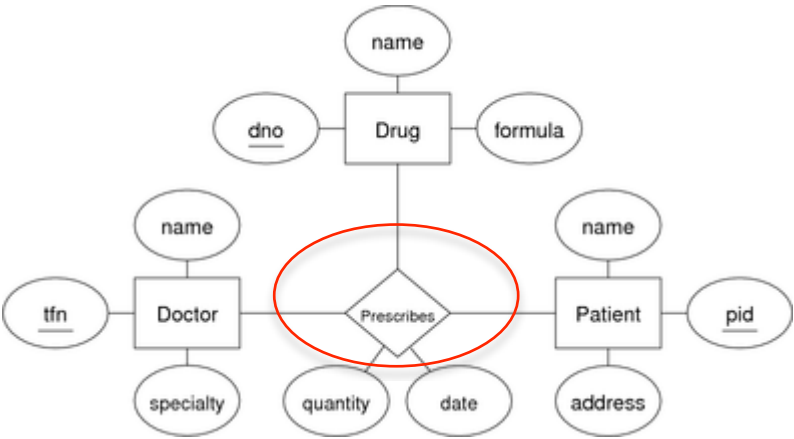
Some people advocate converting n-way relationships into:

- a new entity, and a set of *n* binary relationships

Exercise: 3-way relationship

3/28

Translate the following ER design to a relational schema:



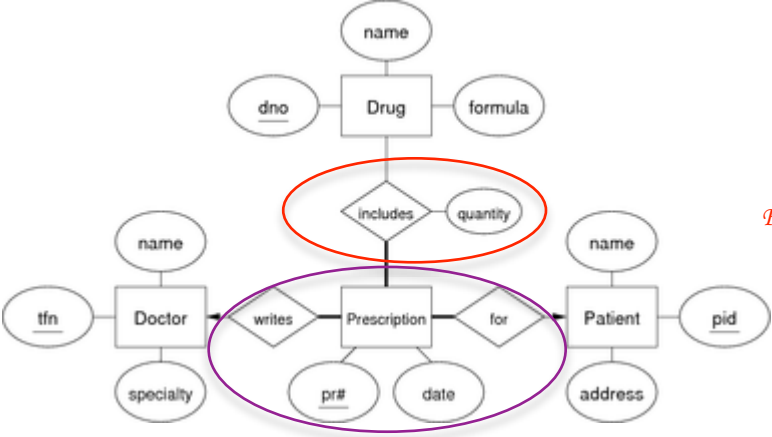
For prescribes, you need a separate table

[Solution]

Exercise: Alternative prescription model

4/28

Translate the following ER design to a relational schema:



For this part, you' d better build a separate table

In this table, you do not need Drug in it

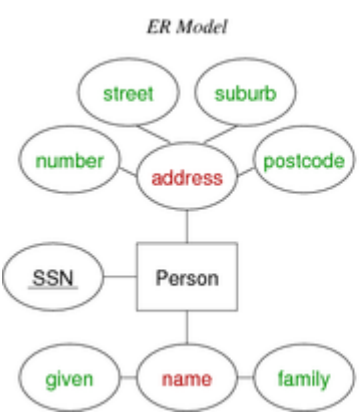
[Solution]

Mapping Composite Attributes

5/28

Composite attributes are mapped by concatenation or flattening.

Example:



Relational Version #1

Person

SSN	name	address
-----	------	---------

504/5 Florence Eastlake 2018

It is slow, but you will find the answer.
As long as you write it, you can use all of the stuff in this string to do searching

Relational Version #2

Person				
SSN	givenName	familyName	...	
...	number	street	suburb	postcode

504 5 Florence Eastlake 2018

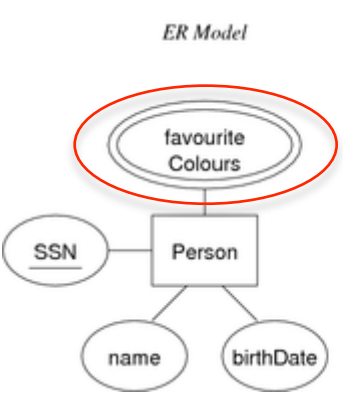
这个的容错率低，但是快

Mapping Multi-valued Attributes (MVAs)

6/28

MVAs are mapped by a new table linking values to their entity.

Example:



Relational Version

Person

SSN	name	birthdate
-----	------	-----------

FavColour

SSN	colour
-----	--------

This is the only way to do Multiple Value
which is
Create a new table for it.

... Mapping Multi-valued Attributes (MVAs)

7/28

Example: the two entities

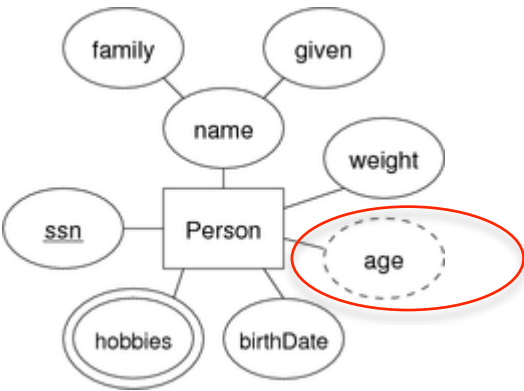
Person(12345, John, 12-feb-1990, [red,green,blue])
Person(54321, Jane, 25-dec-1990, [green,purple])

would be represented as

```
Person(12345, John, 12-feb-1990)
Person(54321, Jane, 25-dec-1990)
FavColour(12345, red)
FavColour(12345, green)
FavColour(12345, blue)
FavColour(54321, green)
FavColour(54321, purple)
```

Exercise: Attribute Mappings

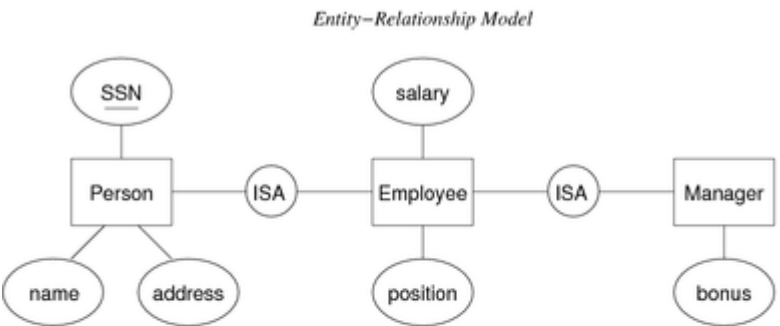
Convert this ER design to relational form:



age
not in any table
we do not need to store it.
What you need is a computation
You can use birthDate to compute it

... Mapping Subclasses

Example of object-oriented mapping:



Relational Model

Puy all the inherent tabel

Person		
SSN	name	address

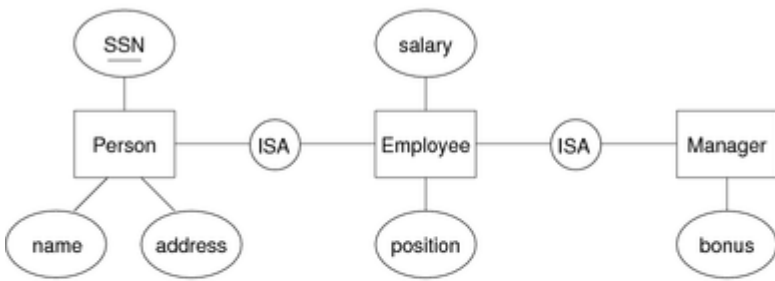
Employee				
SSN	name	address	salary	position

Manager					
SSN	name	address	salary	position	bonus

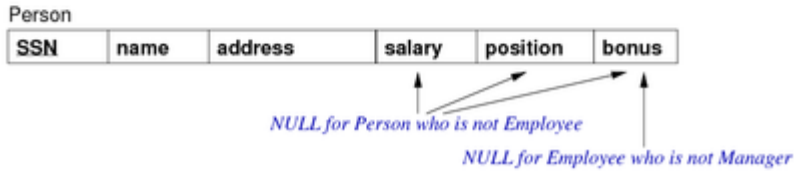
Students love this one

... Mapping Subclasses

Example of single-table-with-nulls mapping:



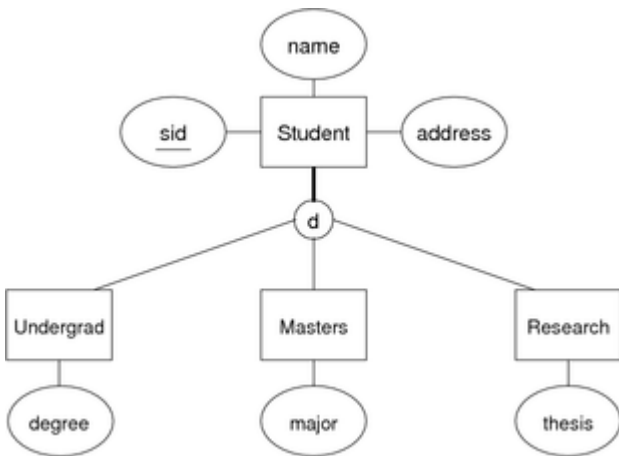
Relational Model



Exercise: Disjoint subclasses

13/28

Translate the following ER design to a relational schema:



Use (a) ER-mapping, (b) OO-mapping, (c) 1-table-mapping

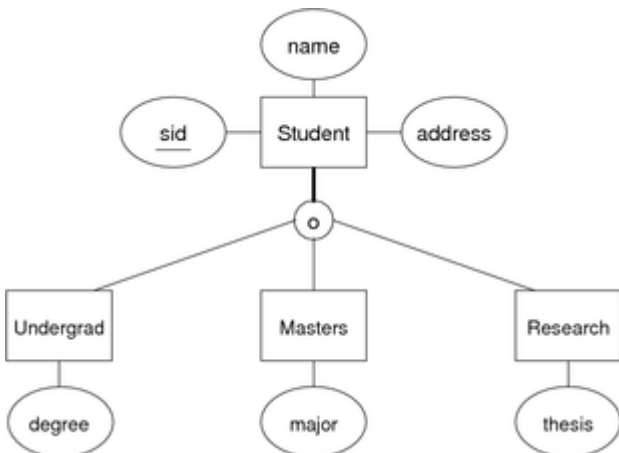
Are there aspects of the ER design that can't be mapped?

[Solution]

Exercise: Overlapping subclasses

14/28

Translate the following ER design to a relational schema:



Use (a) ER-mapping, (b) OO-mapping, (c) 1-table-mapping

Are there aspects of the ER design that can't be mapped?

[\[Solution\]](#)

Relational DBMSs

What is an RDBMS?

16/28

A *relational database management system* (RDBMS) is

- software designed to support large-scale data-intensive applications
- allowing high-level description of data (tables, constraints)
- with high-level access to the data (relational model, SQL)
- providing efficient storage and retrieval (disk/memory management)
- supporting multiple simultaneous users (privilege, protection)
- doing multiple simultaneous operations (transactions, concurrency)
- maintaining reliable access to the stored data (backup, recovery)

Note: databases provide *persistent* storage of information

Describing Data

17/28

RDBMSs implement \approx the relational model.

Provide facilities to define:

- domains, attributes, tuples, tables
- constraints (domain, key, referential)

Variations from the relational model:

- no strict requirement for tables to have keys
- bag semantics, rather than set semantics
- no standard support for general (multi-table) constraints

RDBMS Operations

18/28

RDBMSs typically provide at least the following:

- create/remove a database or a schema
- create/remove/alter tables within a schema
- insert/delete/update tuples within a table
- queries on data, define named queries (views)
- transactional behaviour (ACID)

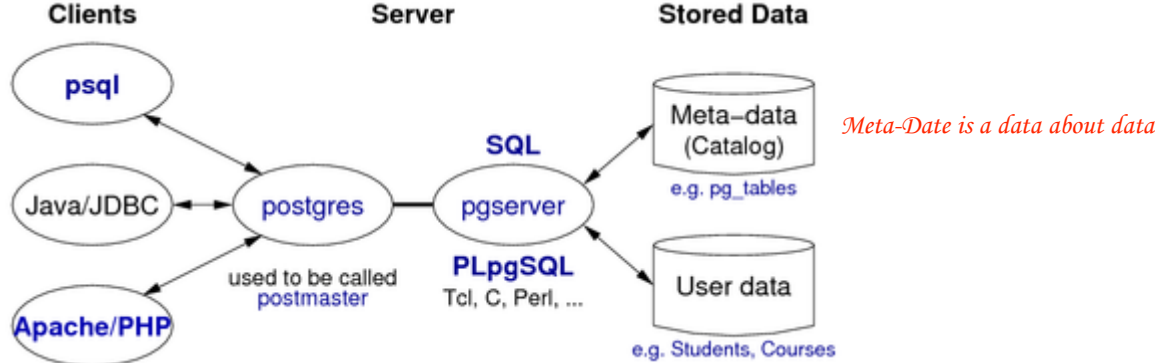
Most also provide mechanisms for

- creating/managing users of the database
- defining/storing procedural code to manipulate data
- implementing complex constraints (triggers)
- defining new data types and operators (less common)

PostgreSQL Architecture

19/28

PostgreSQL's client-server architecture:



Using PostgreSQL

20/28

Using your PostgreSQL server in CSE (once installed):

- login to grieg, set up environment, start server
- use `psql`, etc. to manipulate databases
- stop server, log off grieg

```
wagner$ ssh YOU@grieg
grieg$ priv svr
grieg$ source /svr/YOU/env
grieg$ pg start
grieg$ psql mydb
... do stuff with your database ...
grieg$ pg stop
grieg$ exit
```

... Using PostgreSQL

21/28

PostgreSQL files (helps to understand state of server)

- PostgreSQL home directory ... `/svr/YOU/pgsql1903/`
- under the home directory ...
 - `postgresql.conf` ... main configuration file
 - `base/` ... subdirectories containing database files
 - `postmaster.pid` ... process ID of server process
 - `.s.PGSQL.5432` ... socket for clients to connect to server
 - `.s.PGSQL.5432.lock` ... lock file for socket
- PostgreSQL environment settings ... `/svr/YOU/env`

Building/Maintaining Databases

Managing Databases

23/28

Shell commands:

- **createdb** *dbname*
- **dropdb** *dbname*

(If no *dbname* supplied, assumes a database called *YOU*)

SQL statements:

- **CREATE DATABASE** *dbname*
- **DROP DATABASE** *dbname*

... Managing Databases

24/28

Shell commands (dump/restore):

- **pg_dump** *dbname* > *dumpfile*
- **psql** *dbname* -f *dumpfile*

(Database *dbname* is typically created just before restore)

SQL statements (used in *dumpfile*):

- **CREATE TABLE** *table* (*Attributes+Constraints*)
- **ALTER TABLE** *table* *TableSchemaChanges*
- **COPY** *table* (*AttributeNames*) **FROM STDIN**

Managing Tables

25/28

SQL statements:

- **ALTER TABLE** *table* *TableSchemaChanges*
- **DROP TABLE** *table(s)* [**CASCADE**]
- **TRUNCATE TABLE** *table(s)* [**CASCADE**]

(All conform to SQL standard, but all also have extensions)

DROP . . CASCADE drops objects which depend on the table

TRUNCATE . . CASCADE truncates tables which refer to the table

Managing Tuples

26/28

SQL statements:

- **INSERT INTO** *table* (*attrs*) **VALUES** *tuple(s)*
- **DELETE FROM** *table* **WHERE** *condition*
- **UPDATE** *table* **SET** *AttrValueChanges* **WHERE** *condition*

AttrValueChanges is a comma-separated list of:

- *attrname* = *expression*

Each list element assigns a new value to a given attribute.

Exercise: Generating IDs

27/28

Consider the following schema:

```
create table T (  
    id serial primary key,  
    x integer,  
    y varchar(10)  
);
```

- what does **serial** actually produce (look in the catalog)?
- write INSERT statements to add some tuples

- how could an application program get the generated `id`?
(`select max(id) from T` may not give the correct result; why not?)
-

Managing Other DB Objects

28/28

Databases contain objects other than tables and tuples:

- views, functions, sequences, types, indexes, roles, ...

Most have SQL statements for:

- **CREATE** *ObjectType name ...*
- **DROP** *ObjectType name ...*

Views and functions also have available:

- **CREATE OR REPLACE** *ObjectType name ...*

See PostgreSQL documentation Section IV, Chapter I for SQL statement details.
