COMP9311 Week 03 Lecture

|  |  |
| --- | --- |
| **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:

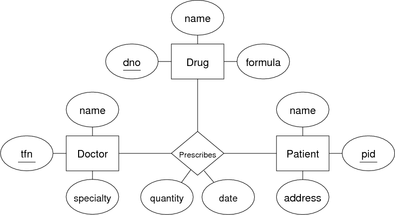
* *n:m* generalises naturally to *n:m:p:q*
  + include foreign key for each participating entity
  + include any other attributes of the relationship
* other multiplicities (e.g. *1:n:m*) ...
  + need to be mapped the same as *n:m:p:q*
  + so not quite an accurate mapping of the ER

Some people advocate converting n-way relationships into:

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

|  |  |
| --- | --- |
| **Exercise: 3-way relationship** | 3/28 |

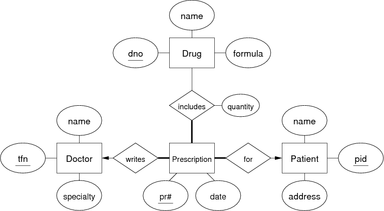
Translate the following ER design to a relational schema:



[[Solution]](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/medical1.sql)

|  |  |
| --- | --- |
| **Exercise: Alternative prescription model** | 4/28 |

Translate the following ER design to a relational schema:

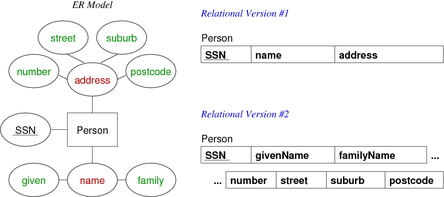


[[Solution]](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/medical2.sql)

|  |  |
| --- | --- |
| **Mapping Composite Attributes** | 5/28 |

Composite attributes are mapped by concatenation or flattening.

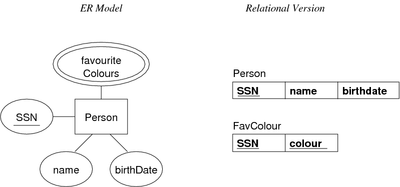
Example:



|  |  |
| --- | --- |
| **Mapping Multi-valued Attributes (MVAs)** | 6/28 |

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

Example:



|  |  |
| --- | --- |
| **... 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)

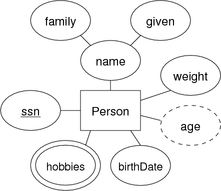
FavColour(12345, blue)

FavColour(54321, green)

FavColour(54321, purple)

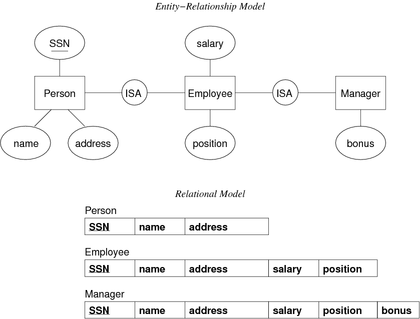
|  |  |
| --- | --- |
| **Exercise: Attribute Mappings** | 8/28 |

Convert this ER design to relational form:



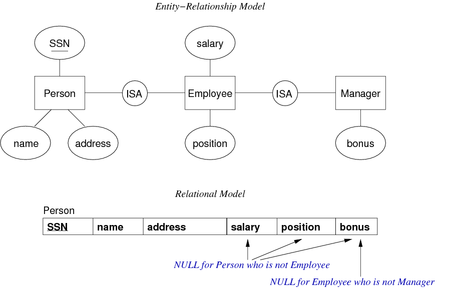
|  |  |
| --- | --- |
| **... Mapping Subclasses** | 11/28 |

[Example of object-oriented mapping:](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/attrs.sql%3E%5BSolution%5D%3C/a%3E%3C/div%3E%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='heading'%3EMapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E9/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EThree%20different%20approaches%20to%20mapping%20subclasses%20to%20tables:%3Cul%3E%3Cli%3E%20ER%20style%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20containing%20attributes%20of%20subclass%20+%20FK%20to%20superclass%20table%3C/ul%3E%3C/small%3E%3Cli%3E%20object-oriented%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20inheriting%20all%20attributes%20from%20all%20superclasses%3C/ul%3E%3C/small%3E%3Cli%3E%20single%20table%20with%20nulls%3Csmall%3E%3Cul%3E%3Cli%3E%20whole%20class%20hierarchy%20becomes%20one%20table,%3Cli%3E%20containing%20all%20attributes%20of%20all%20subclasses%20(null,%20if%20unused)%3C/ul%3E%3C/small%3E%3C/ul%3EWhich%20mapping%20is%20best%20depends%20on%20how%20data%20is%20to%20be%20used.%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='cont'%3E...%20Mapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E10/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EExample%20of%20ER-style%20mapping:%3Cp%3E%3Cdiv%20class='center'%3E%3Cimg%20alt=)



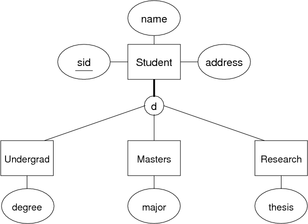
|  |  |
| --- | --- |
| **... Mapping Subclasses** | 12/28 |

[Example of single-table-with-nulls mapping:](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/attrs.sql%3E%5BSolution%5D%3C/a%3E%3C/div%3E%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='heading'%3EMapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E9/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EThree%20different%20approaches%20to%20mapping%20subclasses%20to%20tables:%3Cul%3E%3Cli%3E%20ER%20style%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20containing%20attributes%20of%20subclass%20+%20FK%20to%20superclass%20table%3C/ul%3E%3C/small%3E%3Cli%3E%20object-oriented%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20inheriting%20all%20attributes%20from%20all%20superclasses%3C/ul%3E%3C/small%3E%3Cli%3E%20single%20table%20with%20nulls%3Csmall%3E%3Cul%3E%3Cli%3E%20whole%20class%20hierarchy%20becomes%20one%20table,%3Cli%3E%20containing%20all%20attributes%20of%20all%20subclasses%20(null,%20if%20unused)%3C/ul%3E%3C/small%3E%3C/ul%3EWhich%20mapping%20is%20best%20depends%20on%20how%20data%20is%20to%20be%20used.%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='cont'%3E...%20Mapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E10/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EExample%20of%20ER-style%20mapping:%3Cp%3E%3Cdiv%20class='center'%3E%3Cimg%20alt=)



|  |  |
| --- | --- |
| **Exercise: Disjoint subclasses** | 13/28 |

[Translate the following ER design to a relational schema:](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/attrs.sql%3E%5BSolution%5D%3C/a%3E%3C/div%3E%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='heading'%3EMapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E9/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EThree%20different%20approaches%20to%20mapping%20subclasses%20to%20tables:%3Cul%3E%3Cli%3E%20ER%20style%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20containing%20attributes%20of%20subclass%20+%20FK%20to%20superclass%20table%3C/ul%3E%3C/small%3E%3Cli%3E%20object-oriented%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20inheriting%20all%20attributes%20from%20all%20superclasses%3C/ul%3E%3C/small%3E%3Cli%3E%20single%20table%20with%20nulls%3Csmall%3E%3Cul%3E%3Cli%3E%20whole%20class%20hierarchy%20becomes%20one%20table,%3Cli%3E%20containing%20all%20attributes%20of%20all%20subclasses%20(null,%20if%20unused)%3C/ul%3E%3C/small%3E%3C/ul%3EWhich%20mapping%20is%20best%20depends%20on%20how%20data%20is%20to%20be%20used.%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='cont'%3E...%20Mapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E10/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EExample%20of%20ER-style%20mapping:%3Cp%3E%3Cdiv%20class='center'%3E%3Cimg%20alt=)



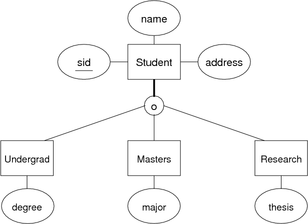
[Use (a) ER-mapping, (b) OO-mapping, (c) 1-table-mapping](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/attrs.sql%3E%5BSolution%5D%3C/a%3E%3C/div%3E%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='heading'%3EMapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E9/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EThree%20different%20approaches%20to%20mapping%20subclasses%20to%20tables:%3Cul%3E%3Cli%3E%20ER%20style%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20containing%20attributes%20of%20subclass%20+%20FK%20to%20superclass%20table%3C/ul%3E%3C/small%3E%3Cli%3E%20object-oriented%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20inheriting%20all%20attributes%20from%20all%20superclasses%3C/ul%3E%3C/small%3E%3Cli%3E%20single%20table%20with%20nulls%3Csmall%3E%3Cul%3E%3Cli%3E%20whole%20class%20hierarchy%20becomes%20one%20table,%3Cli%3E%20containing%20all%20attributes%20of%20all%20subclasses%20(null,%20if%20unused)%3C/ul%3E%3C/small%3E%3C/ul%3EWhich%20mapping%20is%20best%20depends%20on%20how%20data%20is%20to%20be%20used.%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='cont'%3E...%20Mapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E10/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EExample%20of%20ER-style%20mapping:%3Cp%3E%3Cdiv%20class='center'%3E%3Cimg%20alt=)

[Are there aspects of the ER design that can't be mapped?](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/attrs.sql%3E%5BSolution%5D%3C/a%3E%3C/div%3E%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='heading'%3EMapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E9/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EThree%20different%20approaches%20to%20mapping%20subclasses%20to%20tables:%3Cul%3E%3Cli%3E%20ER%20style%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20containing%20attributes%20of%20subclass%20+%20FK%20to%20superclass%20table%3C/ul%3E%3C/small%3E%3Cli%3E%20object-oriented%3Csmall%3E%3Cul%3E%3Cli%3E%20each%20entity%20becomes%20a%20separate%20table,%3Cli%3E%20inheriting%20all%20attributes%20from%20all%20superclasses%3C/ul%3E%3C/small%3E%3Cli%3E%20single%20table%20with%20nulls%3Csmall%3E%3Cul%3E%3Cli%3E%20whole%20class%20hierarchy%20becomes%20one%20table,%3Cli%3E%20containing%20all%20attributes%20of%20all%20subclasses%20(null,%20if%20unused)%3C/ul%3E%3C/small%3E%3C/ul%3EWhich%20mapping%20is%20best%20depends%20on%20how%20data%20is%20to%20be%20used.%3Cp%3E%3Chr%3E%3Cp%3E%3Ctable%20width='100%25'%20cellpadding='0'%3E%3Ctr%20valign='top'%3E%3Ctd%20align='left'%3E%3Cspan%20class='cont'%3E...%20Mapping%20Subclasses%3C/span%3E%3C/td%3E%3Ctd%20align='right'%3E%3Csmall%3E10/28%3C/small%3E%3C/td%3E%3C/tr%3E%3C/table%3E%3Cp%3EExample%20of%20ER-style%20mapping:%3Cp%3E%3Cdiv%20class='center'%3E%3Cimg%20alt=)

[[Solution]](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/subclasses.sql)

|  |  |
| --- | --- |
| **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]](http://www.cse.unsw.edu.au/~cs9311/16s2/lectures/week03/exercises/subclasses.sql)

|  |  |
| --- | --- |
| 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 *≅* 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:

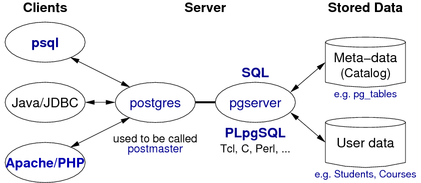
* create/remove a database or a schema
* create/remove/alter tables within a schema
* insert/delete/update tuples within a table
* queries on data, define named queries (views)
* transactional behaviour (ACID)

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 srvr

grieg$ source /srvr/*YOU*/env

grieg$ pg start

grieg$ psql mydb

grieg$ pg stop

grieg$ exit

|  |  |
| --- | --- |
| **... Using PostgreSQL** | 21/28 |

PostgreSQL files (helps to understand state of server)

* PostgreSQL home directory ... /srvr/*YOU*/pgsql903/
* under the home directory ...
  + postgresql.conf ... main configuration file
  + base/ ... subdirectoriess 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 ... /srvr/*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*

(Neither of the above is SQL-standard)

|  |  |
| --- | --- |
| **... 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.

Produced: 9 Aug 2016