COMP3311 Week 02 Lectures

Relational Data Model

1/47

The *relational data model* describes the world as

- a collection of inter-connected *relations* (or *tables*)

The relational model has one structuring mechanism: *relations*

- relations are used to model both entities and relationships

Each *relation* (denoted *R,S,T,...*) has:

- a *name* (unique within a given database)
- a set of *attributes* (which can be viewed as column headings)

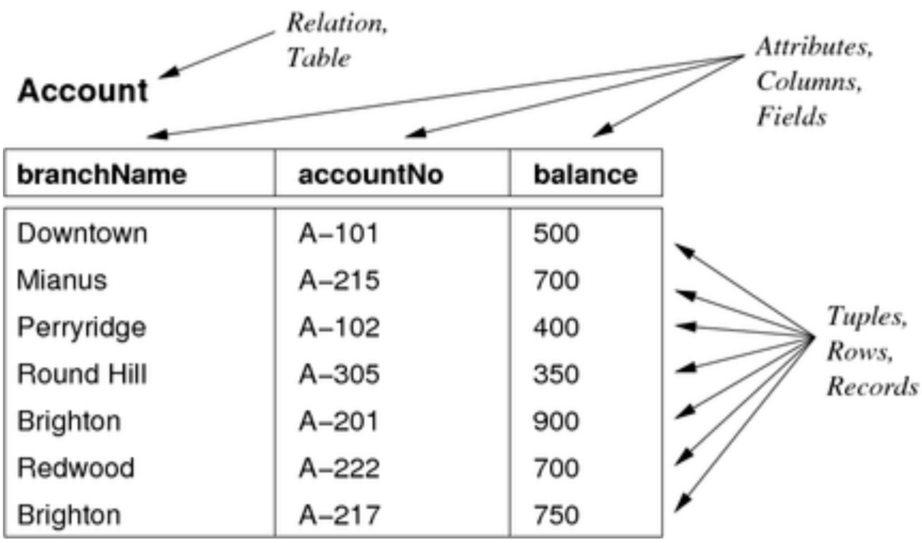
Each *attribute* (denoted *A,B,...* or $a_1,a_2,...$) has:

- a *name* (unique within a given relation)
- an associated *domain* (set of allowed values)

... Relational Data Model

2/47

Example relation (bank accounts):



... Relational Data Model

3/47

Consider relation *R* with attributes $a_1, a_2, \ldots a_n$

Relation schema of *R* : $\mathbf{R}(a_1{:}D_1, a_2{:}D_2, \ldots a_n{:}D_n)$

Tuple of *R* : an element of $D_1 \times D_2 \times \ldots \times D_n$ (i.e. list of values)

Instance of *R* : subset of $D_1 \times D_2 \times \ldots \times D_n$ (i.e. set of tuples)

Note: tuples: (2,3) ≠ (3,2) relation: { (a,b), (c,d) } = { (c,d), (a,b) }

Domains are comprised of atomic values (e.g. integer, string, date)

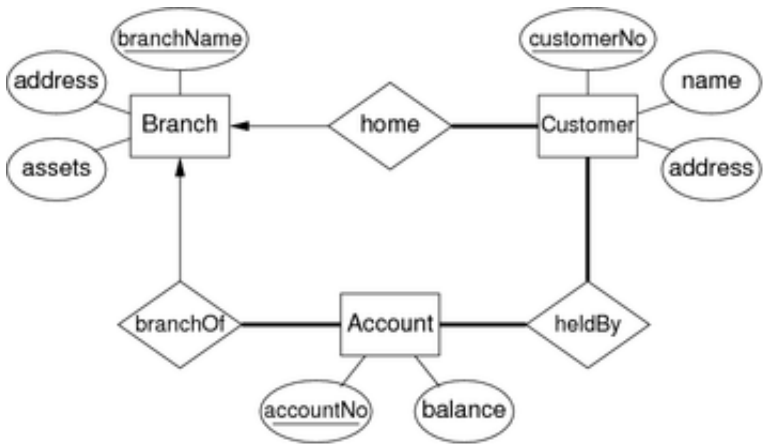
A distinguished value `NULL` belongs to all domains

Each relation has a *key* (subset of attributes unique for each tuple)

Example Database Schema

4/47

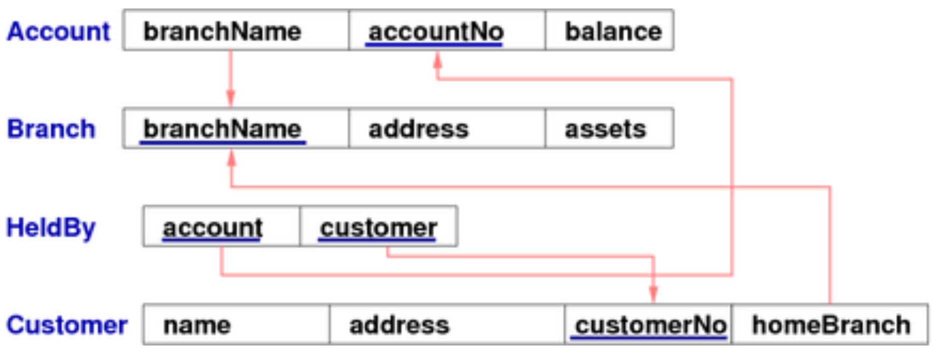
Consider the following ER data model:



... Example Database Schema

5/47

Relational schema derived from this ER model:



Note: distinguish attributes via e.g. `Branch.address` vs `Customer.address`

Example Database (Instance)

6/47

Account

branchName	accountNo	balance
Downtown	A–101	500
Mianus	A–215	700
Perryridge	A–102	400
Round Hill	A–305	350
Brighton	A–201	900
Redwood	A–222	700

Branch

branchName	address	assets
Downtown	Brooklyn	9000000
Redwood	Palo Alto	2100000
Perryridge	Horseneck	1700000
Mianus	Horseneck	400000
Round Hill	Horseneck	8000000
North Town	Rye	3700000
Brighton	Brooklyn	7100000

Customer

name	address	customerNo	homeBranch
Smith	Rye	1234567	Mianus
Jones	Palo Alto	9876543	Redwood
Smith	Brooklyn	1313131	Downtown
Curry	Rye	1111111	Mianus

HeldBy

account	customer
A–101	1313131
A–215	1111111
A–102	1313131
A–305	1234567
A–201	9876543
A–222	1111111
A–102	1234567

Integrity Constraints

7/47

To represent real-world problems, need to describe

- what values are/are not allowed
- what combinations of values are/are not allowed

Constraints are logical statements that do this:

- *domain constraints* limit the set of values that attributes can take
- *key constraints* identify attributes that uniquely identify tuples
- *entity integrity constraints* require keys to be fully-defined
- *referential integrity constraints* require references to other tables to be valid

... Integrity Constraints

8/47

Domain constraints example:

- `Employee.age` attribute is typically defined as `integer`
- better modelled by adding extra constraint `(15<age<66)`

Note: `NULL` satisfies all domain constraints (except (NOT NULL))

Key constraints example:

- `Student( id, ...)` is guaranteed unique
- `Class(...,day,time,location,...)` is unique

Entity integrity example:

- `Class(...,Mon,2pm,Lyre,...)` is well-defined
- `Class(...,`**NULL**`,2pm,Lyre,...)` is not well-defined

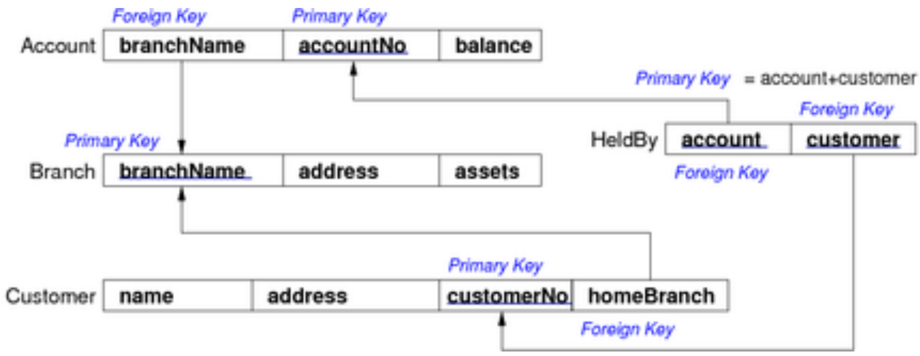
Referential Integrity

9/47

Referential integrity constraints

- describe references between relations (tables)
- are related to notion of a *foreign key* (FK)

Example:



... Referential Integrity

10/47

A set of attributes F in relation R_1 is a *foreign key* for R_2 if:

- the attributes in F correspond to the primary key of R_2
- the value for F in each tuple of R_1
    - either occurs as a primary key in R_2
    - or is entirely `NULL`

Foreign keys are critical in relational DBs; they provide ...

- the "glue" that links individual relations (tables)
- the way to assemble query answers from multiple tables
- the relational representation of ER relationships

Relational Databases

11/47

A *relational database schema* is

- a set of relation schemas $\{ R_1, \ R_2, \ ... \ R_n \}$, and
- a set of integrity constraints

A *relational database instance* is

- a set of relation instances $\{ r_1(R_1), \ r_2(R_2), \ ... \ r_n(R_n) \}$
- where all of the integrity constraints are satisfied

One of the important functions of a relational DBMS:

- ensure that all data in the database satisfies constraints

Changes to the data fail if they would cause constraint violation

---

Describing Relational Schemas

12/47

We need a formalism to express relational schemas
(which is more detailed than boxes-and-arrows diagrams used above)

SQL provides a *D*ata *D*efinition *L*anguage (DDL) for this.

```
CREATE TABLE TableName (
    attrName1 domain1 constraints1 ,
    attrName2 domain2 constraints2 ,
    ...
    PRIMARY KEY (attri,attrj,...)
    FOREIGN KEY (attrx,attry,...)
                REFERENCES
                OtherTable (attrm,attrn,...)
);
```

SQL Syntax in a Nutshell

13/47

Comments: everything after `--` *is a comment*

Identifiers: alphanumeric (a la C), but also `"An Identifier"`

Reserved words: many e.g. `CREATE`, `SELECT`, `TABLE`, ...

Strings: e.g. `'a string'`, `'don''t ask'`, but no `'\n'`

Numbers: like C, e.g. `1`, `-5`, `3.14159`, ...

Identifiers and reserved words are case-insensitive:

- `TableName` = `tablename` = `TaBLeNamE` != `"TableName"`

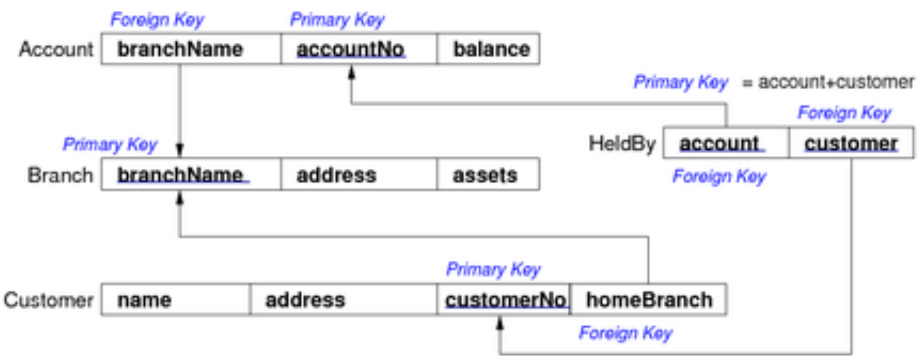
Types: `integer`, `float`, `char(`*n*`)`, `varchar(`*n*`)`, `date`, ...

Operators: `=`, `<>`, `<`, `<=`, `>`, `>=`, `AND`, `OR`, `NOT`, ...

Exercise 1: Simple Relational Schema

14/47

Express the following schema in SQL DDL:



Assume only two domains: `text` and `integer`.

Augment the previous schema to enforce:

- no accounts can be overdrawn
- customer numbers are seven-digit integers
- account numbers look like `A-101`, `B-306`, etc.
- the assets of a branch is the sum of the balances in all of the accounts held at that branch

Assume that all standard SQL types (domains) are available.

Add new domain to define account numbers and use it throughout

Mapping ER Designs to Relational Schemas

ER to Relational Mapping 17/47

One useful strategy for database design:

- perform initial data modelling using ER
  (conceptual-level modelling)
- transform conceptual design into relational model
  (implementation-level modelling)

A formal mapping exists for ER model → Relational model.

This maps "structures"; but additional info is needed, e.g.

- concrete domains for attributes and other constraints

Relational Model vs ER Model 18/47

Correspondences between relational and ER data models:

- attribute(ER) ≅ attribute(Rel), entity(ER) ≅ tuple(Rel)
- entity set(ER) ≅ relation(Rel), relationship(ER) ≅ relation(Rel)

Differences between relational and ER models:

- Rel uses relations to model entities *and* relationships
- Rel has no composite or multi-valued attributes (only atomic)
- Rel has no object-oriented notions (e.g. subclasses, inheritance)

Mapping Strong Entities 19/47

An entity set E with atomic attributes $a_1, a_2, ... a_n$

maps to

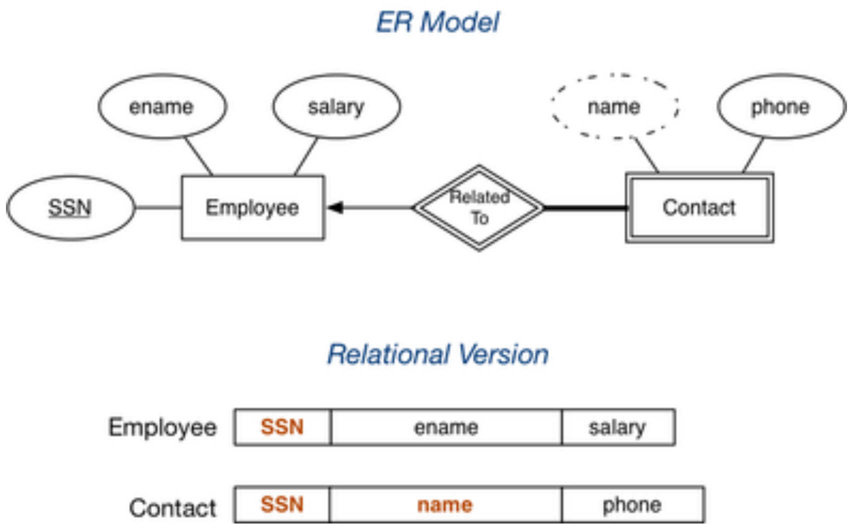
A relation R with attributes (columns) $a_1, a_2, ... a_n$

Example:

Note: the key is preserved in the mapping.

Mapping Weak Entities 20/47

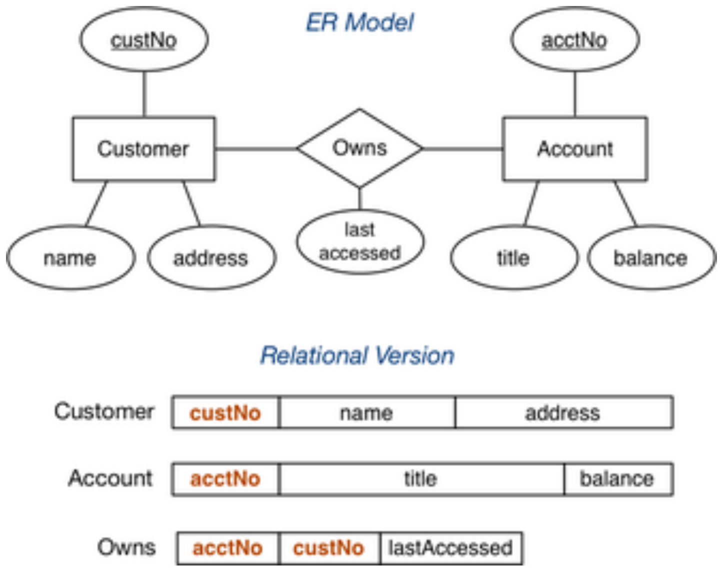
Example:



Mapping N:M Relationships

21/47

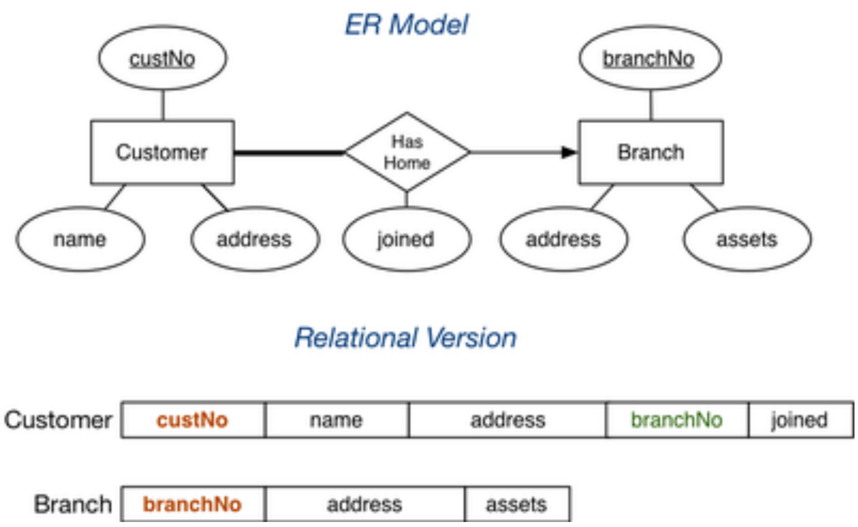
Example:



Mapping 1:N Relationships

22/47

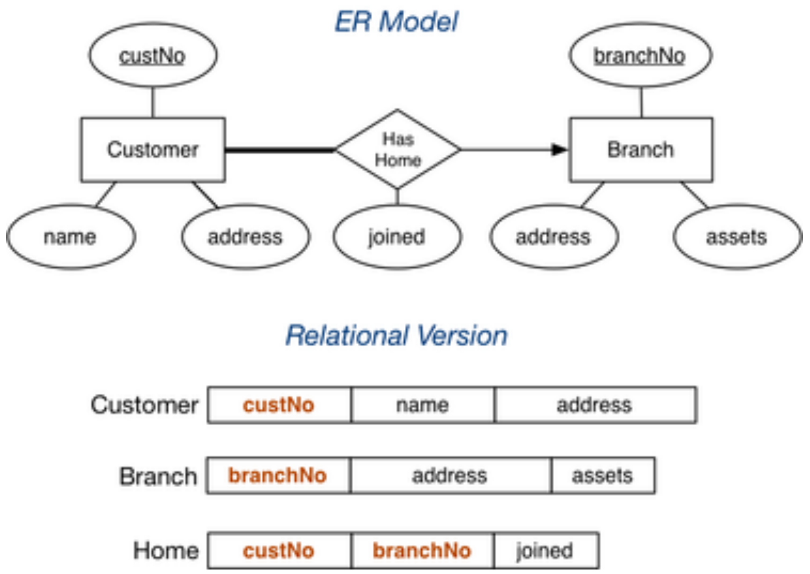
Example:



Exercise 2: Mapping 1:N Relationships

23/47

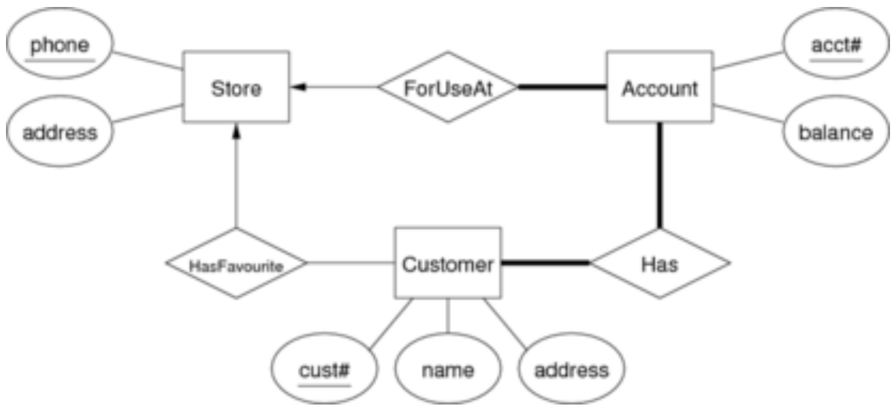
Would this be a suitable mapping for a 1:N relationship?



Exercise 3: ER-to-Relational (1)

24/47

Convert this ER design to relational form:

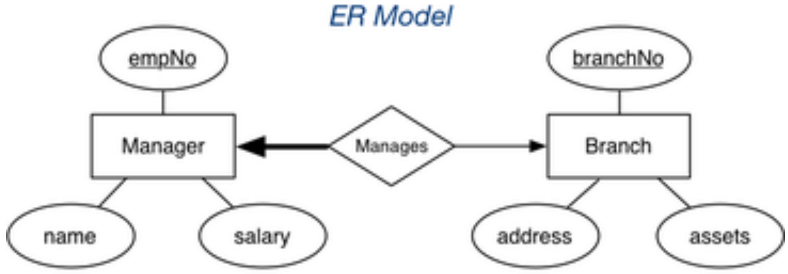


Mapping 1:1 Relationships

25/47

Example:

ER Model



Relational Version

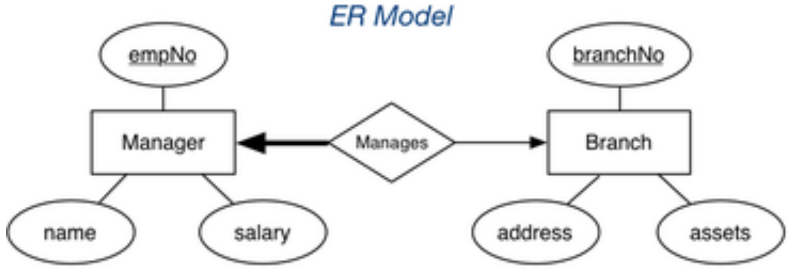
Manager	**empNo**	name	salary	branchNo
Branch	**branchNo**	address	assets	

Exercise 4: Mapping 1:1 Relationships

26/47

What are advantages/disadvantages of this mapping:

ER Model



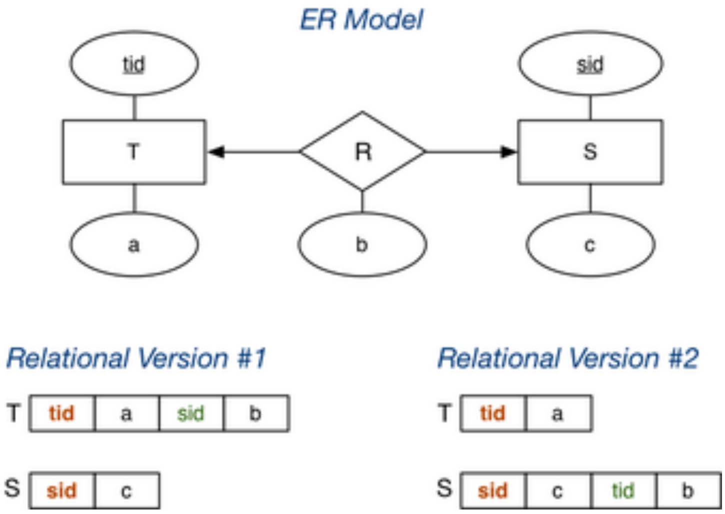
Relational Version

Manager	**empNo**	name	salary	
Branch	**branchNo**	address	assets	empNo

... Mapping 1:1 Relationships

27/47

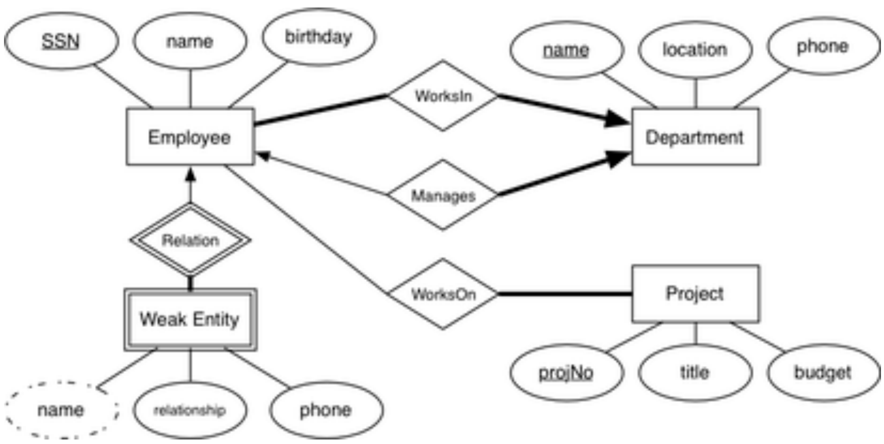
If there is no reason to favour one side of the relationship ...



Exercise 5: ER-to-Relational (2)

28/47

Convert this ER design to relational form:



Mapping n-way Relationships

29/47

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

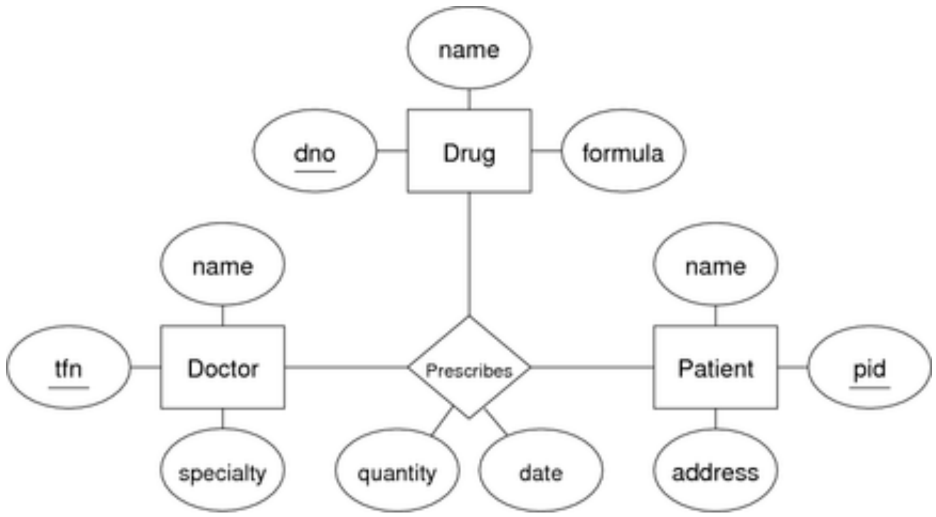
Some people advocate converting n-way relationships into:

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

Exercise 6: 3-way relationship

30/47

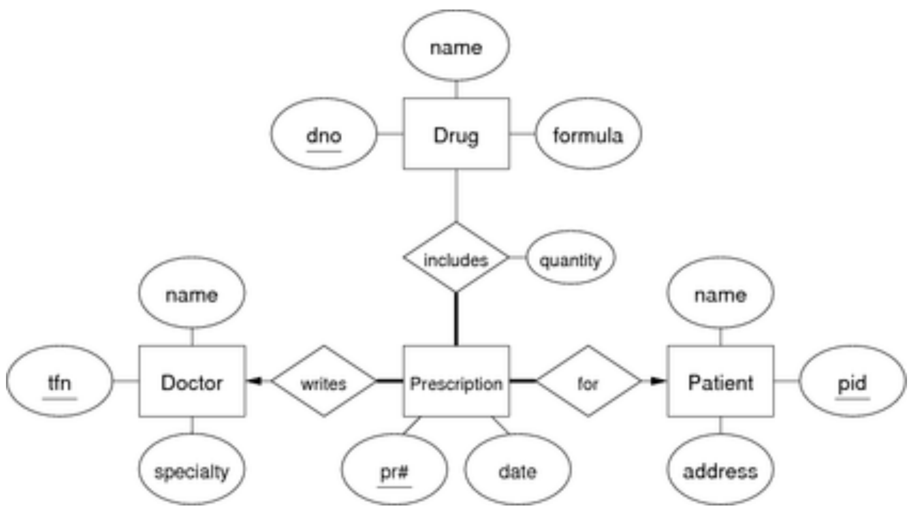
Translate the following ER design to a relational schema:



Exercise 7: Alternative prescription model

31/47

Translate the following ER design to a relational schema:

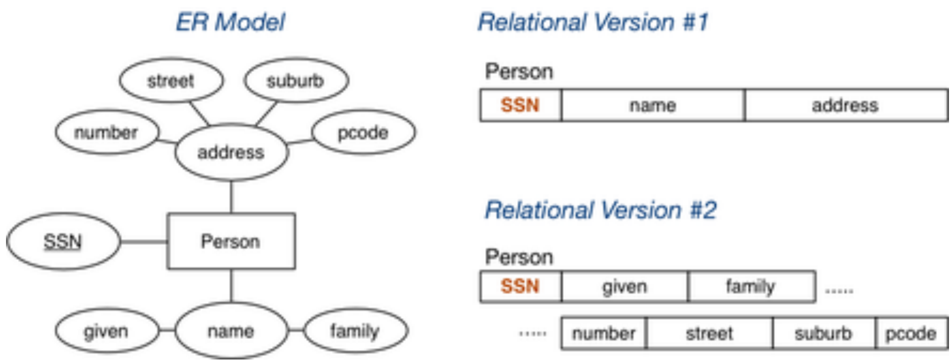


Mapping Composite Attributes

32/47

Composite attributes are mapped by concatenation or flattening.

Example:

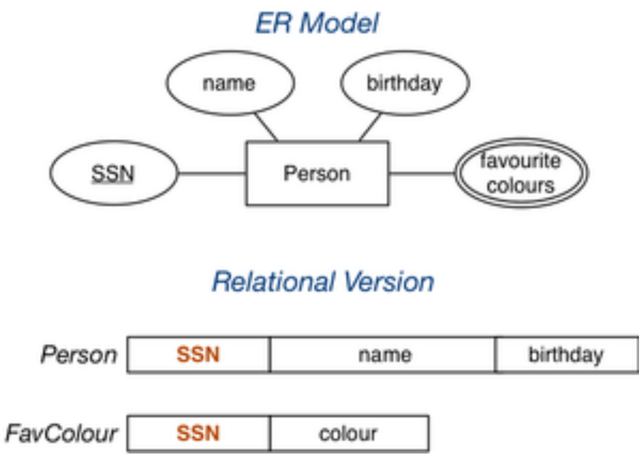


Mapping Multi-valued Attributes (MVAs)

33/47

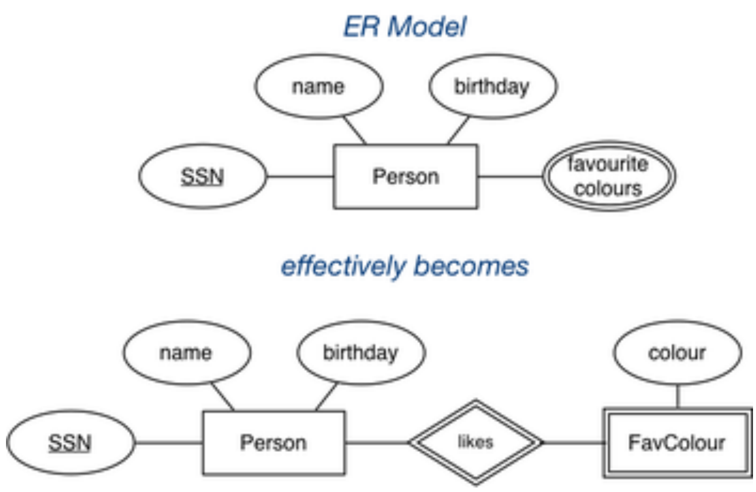
MVAs are mapped by a new table linking values to their entity.

Example:



... Mapping Multi-valued Attributes (MVAs)

34/47



... Mapping Multi-valued Attributes (MVAs)

35/47

Example: the two entities

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

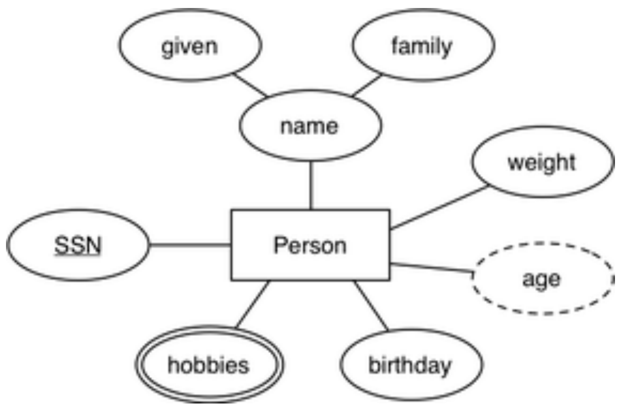
would be represented as

```
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 8: Attribute Mappings

36/47

Convert this ER design to relational form:



Mapping Subclasses

37/47

Three different approaches to mapping subclasses to tables:

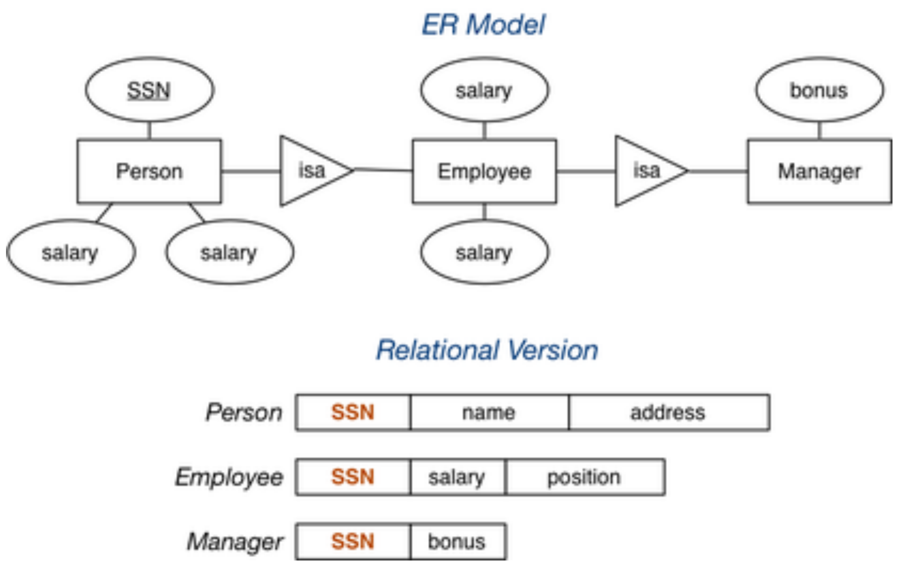
- ER style
  - each entity becomes a separate table,
  - containing attributes of subclass + FK to superclass table
- object-oriented
  - each entity becomes a separate table,
  - inheriting all attributes from all superclasses
- single table with nulls
  - whole class hierarchy becomes one table,
  - containing all attributes of all subclasses (null, if unused)

Which mapping is best depends on how data is to be used.

... Mapping Subclasses

38/47

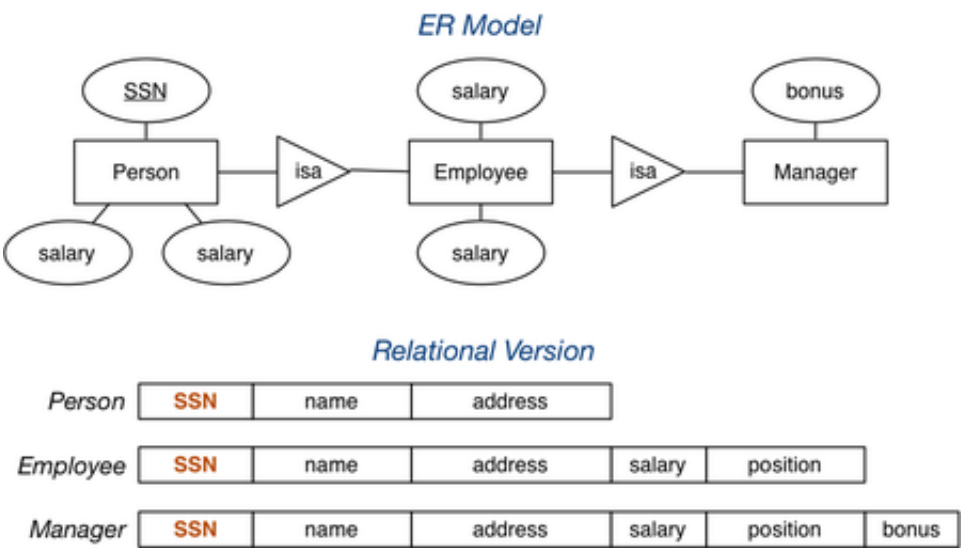
Example of ER-style mapping:



... Mapping Subclasses

39/47

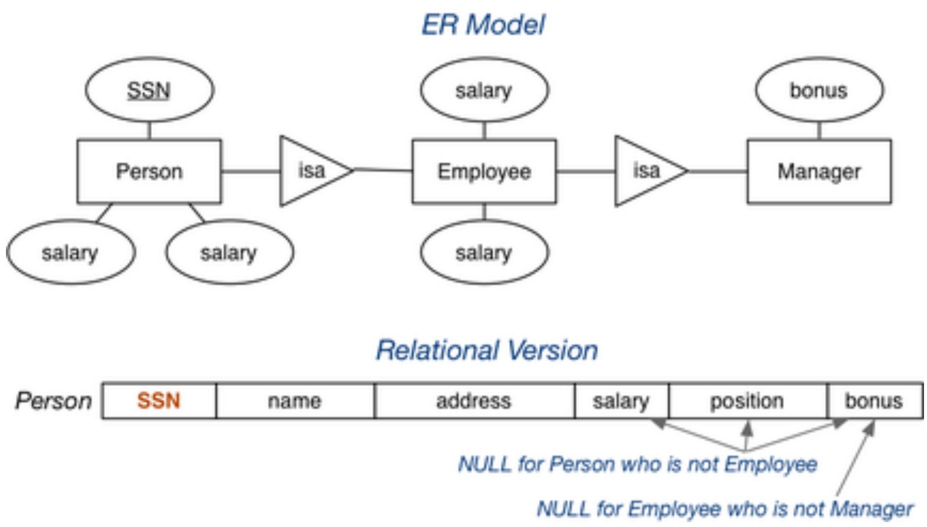
Example of object-oriented mapping:



... Mapping Subclasses

40/47

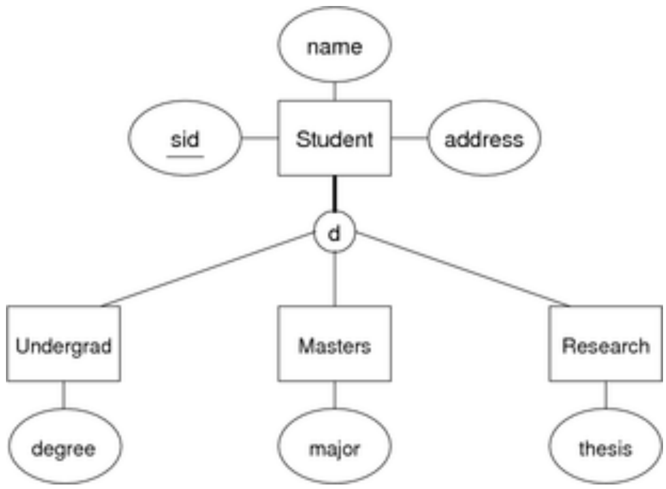
Example of single-table-with-nulls mapping:



Exercise 9: Disjoint subclasses

41/47

Translate the following ER design to a relational schema:



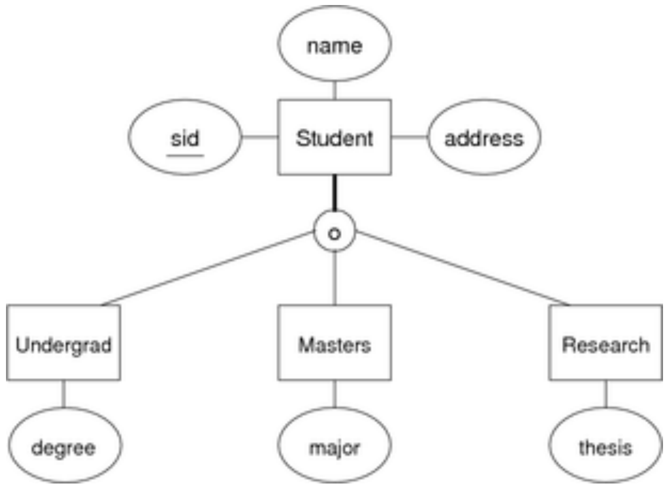
Use (a) ER-mapping, (b) OO-mapping, (c) 1-table-mapping

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

Exercise 10: Overlapping subclasses

42/47

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?

Relational DBMSs

What is an RDBMS?

44/47

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

45/47

RDBMSs implement ≅ the relational model.

Provide facilities to define:

- attributes, tuples, relations/tables
- domains (built-in and user-defined)
- 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

46/47

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)

RDBMSs in COMP3311

47/47

Modern RDBMSs

- support most of SQL standard (plus extensions)
- store catalog (meta-data) in `information_schema` tables

PostgreSQL

- open-source client-server RDBMS
- has strong extensibility model (new types, new functions, ...)
- flexible concurrency control

SQLite

- open-source serverless embeddable RDBMS
- loose in how typing is applied, allows `"..."`

Produced: 26 Sep 2019