COMP3311 WEEK03 LECTURE

N-WAY RELATIONSHIPS

- N:M generalizes naturally to N:M:P:Q
  - Include foreign key for each participating entity;
  - Include ay other attributes of the relationship
- Other multiplicities (eg. 1:N:M)
  - Need to be mapped the same as N:M:P:Q
  - Not quite an accurate mapping of ER

Exercise 1: 3-way relationship

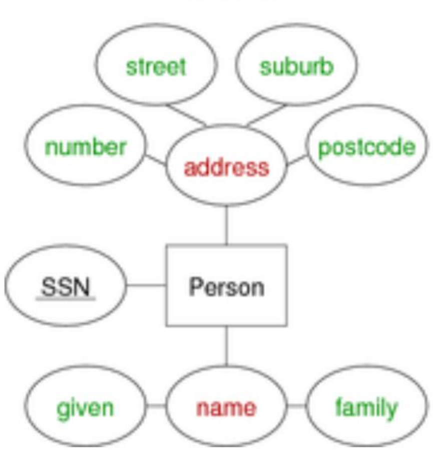
Exercise2: Alternative prescription model

MAPPING COMPOSITE ATTRIBUTES

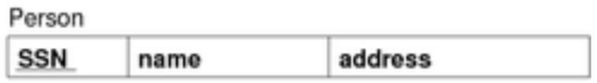
Composite attributes are mapped by concatenation or flattening

Composite attributes are represented by **components**

ER Model



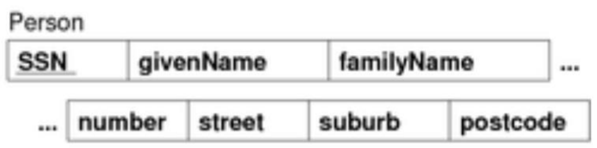
Relational Version #1



Person

SSN	name	address

Relational Version #2



Person

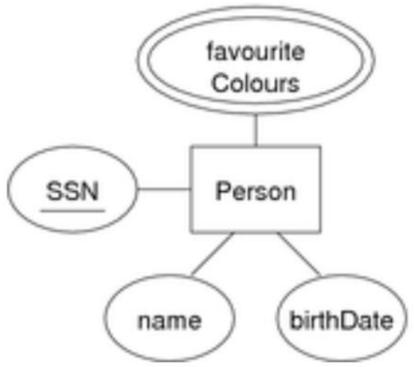
SSN	givenName	familyName	number	street	suburb	postcode

MAPPING MULTI-VALUED ATTRIBUTES (MVAS)

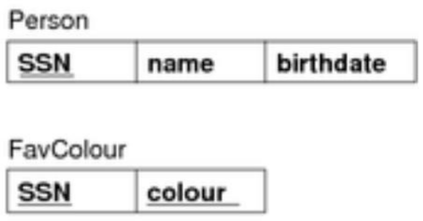
Multi-valued attributes are represented by a **separate table**

MVAS are mapped by new table linking values to their entity.

ER Model



Relational Version



Person

SSN	name	birthdate

FavColour

SSN	colour

MAPPING MULTI-VALUED ATTRIBUTES

Example: the two entities:

Person(12345, John, 12-feb-1990, [red, green,blue])

Person(54321, Jane, 25-dec-1990, [green, purple])

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)

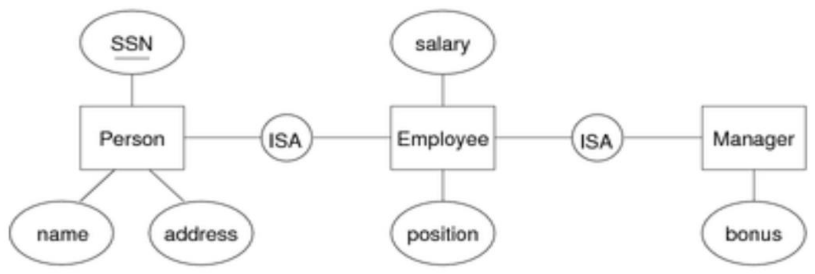
Same as mapping to relational diagram:
MAP the other attributes first
When its multi-valued attribute:
Represent by separate table
And related to the primary key

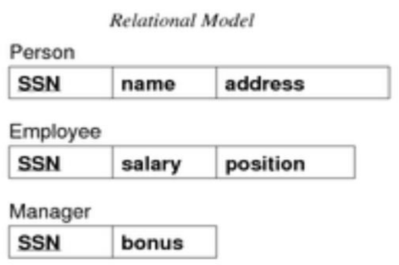
MAPPING SUBCLASSES

- ER
  - Each entity becomes separate table,
  - Containing attributes of subclasses + foreign key to superclass table.
- Object-oriented
  - Each entity becomes a separate table,
  - Inheriting all attributes from all superclass
- Single table with nulls
  - Whole class hierarchy becomes one table,
  - Containing all attributes of all subclasses

1. ER STYLE SUBCLASS MAPPING

Hint: Person IS A Employee, Employee IS A Manager. Which means both employee and manager has to be a person, thus SSN(primary key or person) has to be part of their attributes.





Relational Model

Person

SSN	name	address

Employee

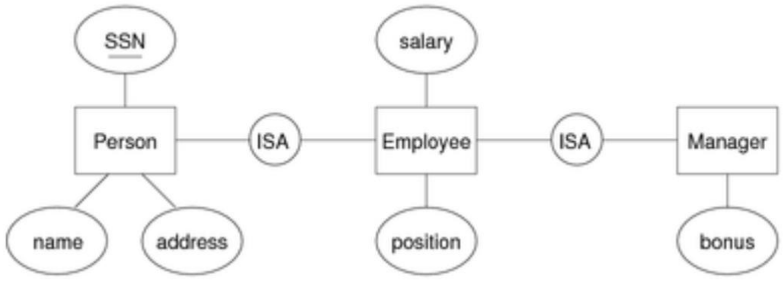
SSN	salary	position

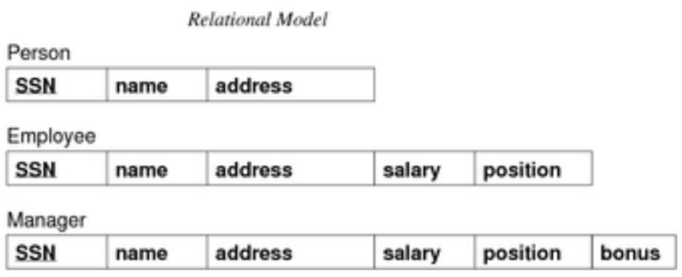
Manager

SSN	bonus

2. OBJECT-ORIENTED MAPPING

Hint: Person IS A Employee, Employee IS A Manager. Which means employee is a person it should include all the attributes from Person, and manager is an employee, so it should contain all the attributes from employee and person.





Relational Model

Person

SSN	name	address

Employee

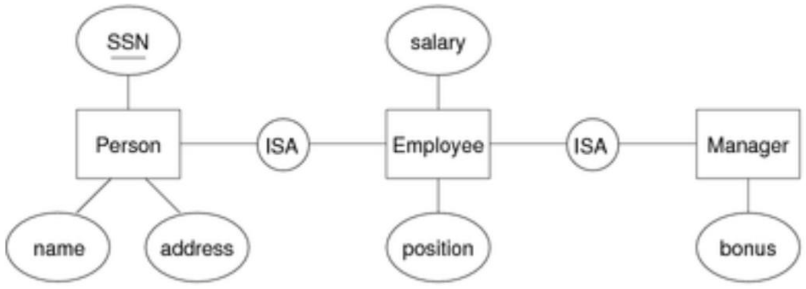
SSN	name	address	salary	position

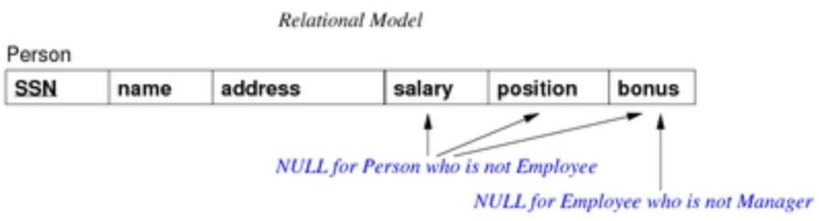
Manager

SSN	name	address	salary	position	bonus

3. SINGLE-TABLE-WITH-NULLS MAPPING

Hint: Both Employee and Manager are Person, so only display a Person table includes all the attributes from all three entities, while someone is not employee or manager, put 'null' value for the specific attributes.





Relational Model

Person

SSN	name	address	salary	position	bonus

NULL for Person who is not Employee
NULL for Employee who is not Manager

RELATIONAL DBMSs

WHAT IS AN RDBMS?

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)

DESCRIBING DATA

RDBMS implement ≅ the RELATIONAL TABLE

Provide facilities to define:

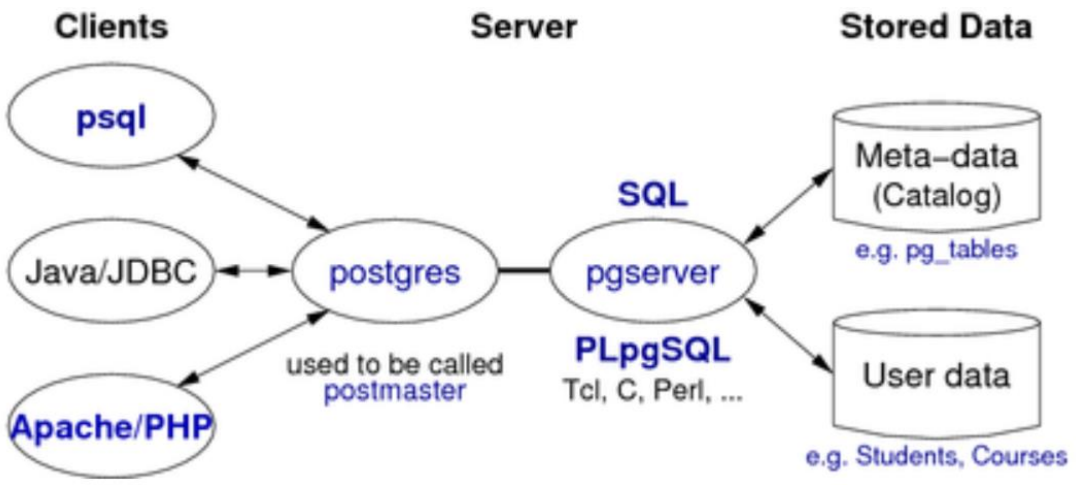
- Domains, attributes, tuples, tables
- Constraints (domain, key, referential)

Variations from the relational model:

- No strict requirement for tables to have keys
- Bag semantics, rather than set semantics
  - The operations of the relational algebra are extended to operate on bags by defining their action on tuple multiplicities.
- No standard support for general (multi-table) constraints

PostgreSQL ARCHETECTURE

PostgreSQL's client-server architecture



- PostgreSQL is a general purpose and object-relational database management system
- It allows you to add custom function developed using different programming languages
- Its designed to be extensible, you can define your own data types, functional languages

MANAGING DATABASES

SHELL COMMANDS:

- Createdb dbname
- Dropdb dbname (remove a PostgreSQL database)
- Pg_dump dbname > dumpfile (extract a PostgreSQL database to a script file or other archive file

  Eg. Pg_dump mydb > mydb.sql)

- Psql dbname -f dumpfile (execute commands from a file on the given database

SQL STATEMENT:

- CREATE DATABASE dbname
- DROP DATABASE dbname
- CREATE TABLE table (attributes + constraints)
- ALTER TABLE table TableSchemaChanges (add, delete or modify columns in an existing table)
- COPY table (attributeNames) FROM STDIN (STDIN – specifies that input comes from the client application; COPY – copy data between a file and a table)

MANAGING TABLES

SQL STATEMENT:

- ALTER TABLE table TableSchemaChanges
- DROP TABLE table(s) [CASCADE] (DROP TABLE – remove a table; [CASCADE] automatically drop objects that depends on the table (such as views))
- TRUNCATE TABLE table(s) [CASCADE] (empty a table or set of tables)

MANAGING TUPLES

SQL STATEMENT:

- INSERT INTO table (attributes) VALUES tuple(s)
- DELETE FROM table WHERE condition
- UODATE 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

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

- SERIAL data type allows you to automatically generate unique integer numbers (ID, identity, auto-increment, sequence) for a column.

```
CREATE TABLE teams (
        Id    SERIAL UNIQUE,
        Name VARCHAR (90)
);

--insert a row, ID will be automatically generated

INSERT INTO teams(name) VALUES ('xxxxxxxx');
--retrieve generated ID (just one of the possible options)
SELECT LASTVAL();

--returns:1
```