Limitations of the Relational Model

- Database systems employing object-oriented models have been touted as the emerging form for the next generation of systems.
- Some say that just as the relational model has supplanted the network (CODASYL) and hierarchical models, so too will object-oriented models supplant the relational model.

Questions:

1. Are these predictions true?
2. What are the reasons behind them?

To begin, we must examine the strengths and weaknesses of the relational model.

Major strengths of the relational model:

- The data model and access to it is simple to understand and use, even for those who are not experienced programmers.
  - The model of data represented in tables is remarkably simple.
  - Access to data via the model does not require *navigation* (roughly, following pointers), as do the CODASYL and network models.
  - It admits a simple (in principle), declarative query language.
- There are straightforward database *design procedures.*
- The data model admits a solid and well-understood mathematical foundation (first-order predicate logic). This has facilitated the development of a sophisticated theoretical underpinning, which has contributed greatly to the features of practical systems.
- Efficient implementation techniques are well known and widely used.
- Standards exist both for query languages (SQL) and for interfaces via programming languages (embedded SQL and ODBC/CLI).

With all of these strengths, why go beyond the relational model in general, and to an object-oriented model in particular?

1. There are some forms of data and knowledge which the relational model cannot accommodate easily and adequately.

2. Object-oriented programming languages are emerging as the dominant form within development environments for large-scale software systems.
    - A relational database model is not a good match to an object-oriented host language.

3. Language-independent system environments which are based upon object-oriented models are emerging, and promise to be extremely important in the future.
    - Dominant example: OMG (Object Management Group) standards.

To begin, the first point is examined in more detail.

Limitations of the Relational Model

It is convenient to divide the limitations up into two categories.

First of all, there are always very special types of data which require special forms of representation and/or inference. Some examples are the following.

Limitations regarding special forms of data:

- Temporal data
- Spatial data
- Multimedia data
- Unstructured data (warehousing/mining)
- Document libraries (digital libraries)

Limitations regarding SQL as the query language:

- Recursive queries (*e.g.,* compute the ancestor relation from the parent relation):
  - Although part of the SQL:1999 standard, recursive queries are still not supported by many systems (*e.g.* PostgreSQL).
  - Support for recursive queries in SQL:1999 is weak in any case. (Only so-called *linear queries* are supported.)

On the other hand, there are also some fundamental shortcomings of the relational model, which are addressed next.

Fundamental Limitations of the Relational Model

1. Object identity:

In entity-relationship modelling, explicit object types, such as *Employee, Department, Project, etc*., are specified. In the relational model, these may survive only as names of relations.

- In the relational model, entities have no independent identification or existence. Objects can only be identified and accessed indirectly via the identification of those attributes which characterize them.

Example:

- In the Company database of the textbook, it is difficult to speak of an employee as a fundamental entity.

- An employee only exists by virtue of a list of attributes in some tables.

2. Explicit relationships:

In entity-relationship modelling, explicit entities and relationships were specified. In the relational model, the identities of relationships have no explicit representation.

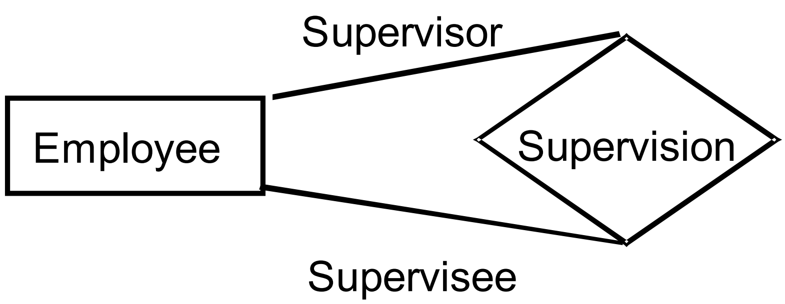
Relationships must be recovered by executing query operations on the database.

These relationships must be known to the user from information not contained in the relational representation.

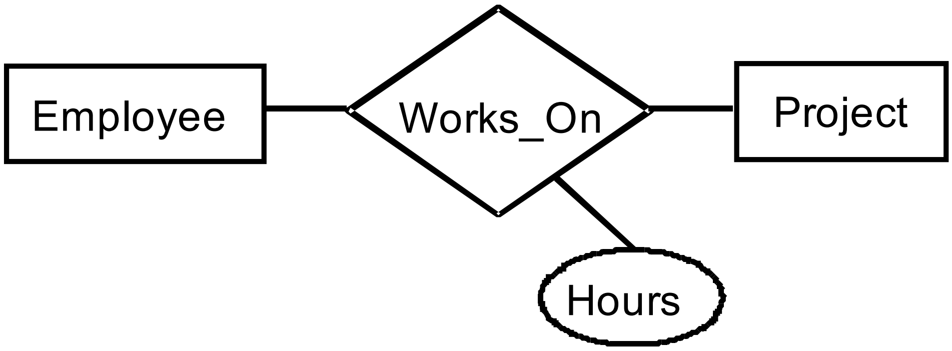
There is a *hidden semantics* in the relational model.

Example from the Company database of the text:

- In the relational realization of the information embodied diagram shown below, the Supervision relation, as well as the Supervisor and Supervisee rôles, are hidden.



- In the following example, the relationship becomes a relation connecting two other relations.

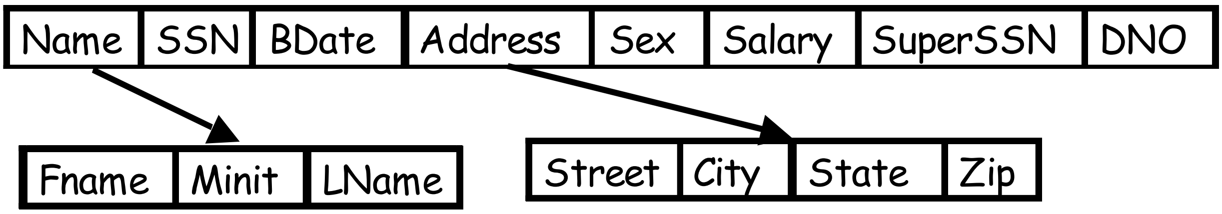


- Thus, the distinction between an entity and a relationship is blurred.

3. Structured data objects:

First normal form (1NF) stipulates that the values for attributes in a tuple be atomic.

This prohibits the kind of complex values illustrated below, in which the values of domains are themselves tuples.



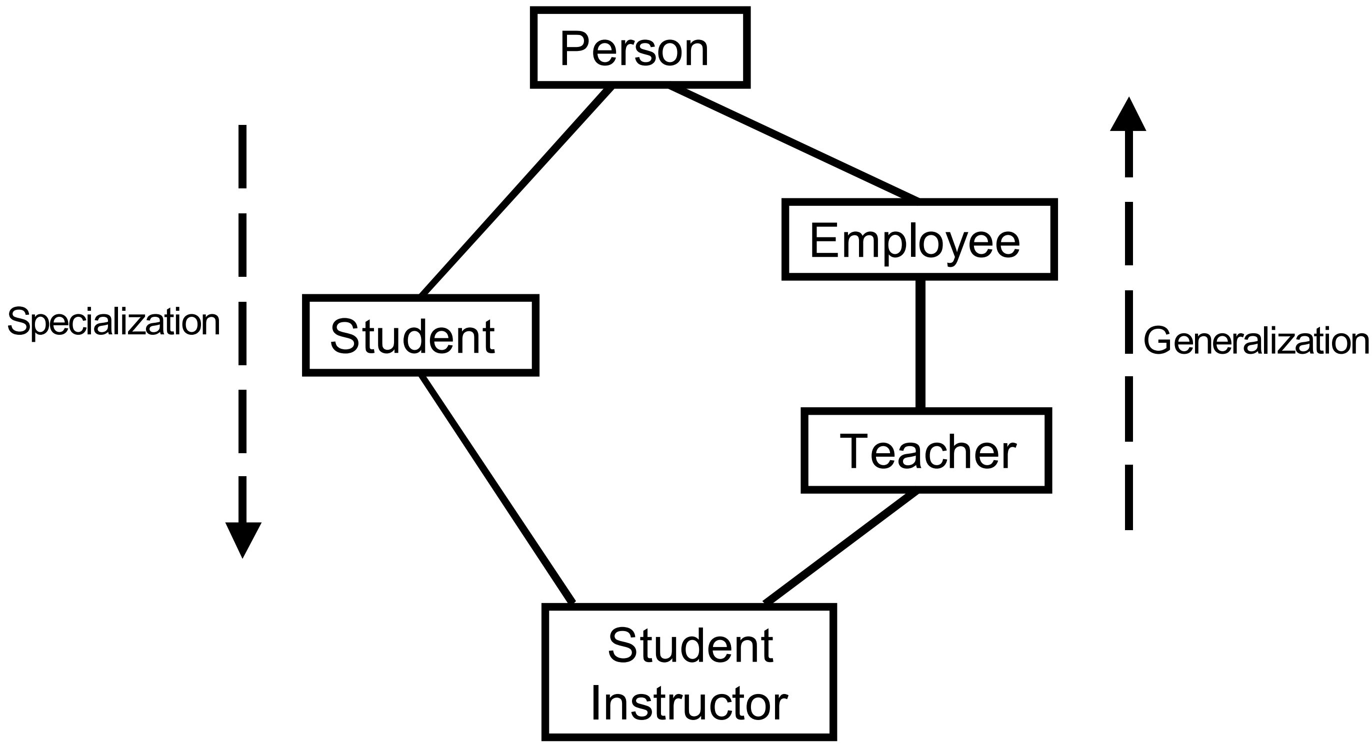
Note that this is a natural form of modelling in this application.

It also prohibits so-called *collection types*, such as sets, lists, and multisets. This is illustrated in the example below, in which a separate relation capturing department locations must be used.

Department				
Dname	Dnumber	MGRSSN	MGR-Startdate	DLocations
Research	5	333445555	1998-05-22	{Bellaire, Sugarland, Houston}
Administration	4	987654321	1995-01-01	Stafford
Headquarters	1	888665555	1981-06-19	Houston

4. Generalization and inheritance:

The classes of entities to be modelled in a database often have a natural hierarchical structure. An example from a university situation is shown below.



Key ideas:

- The class of objects associated with a type higher in the hierarchy is a superset of that associated with that of a class lower in the hierarchy
  - Every Employee is a Teacher.
  - Every Student Instructor is both a Student and an Instructor.
  - For this reason, these are sometimes called *ISA hierarchies*.

- Classes which lie lower in the hierarchy *inherit* attributes from those higher up.

  - Assume that every person has an SSN.
  - Then every Student, Employee, Teacher, and and Student Instructor has an SSN also.

  - Assume that every Employee has an employee number. Then every Teacher and every Student Instructor has an employee number also, but this is not necessarily the case for Persons or Students.

- When an object class inherits attributes from two or more object classes (*e.g.,* Student Instructor from both Student and Instructor), it is called *multiple inheritance.*

Inheritance is not part of the relational model.

5. Methods

Often, it is convenient to record explicitly special queries on a database.

- In the relational model, for read-only queries, this may be accomplished via views (although they introduce overhead, due to the need of the system to maintain the current value).

- For updates, there is no similar mechanism available in the relational model. Such procedures must be maintained outside of the relational model itself.

Example: Add a new employee.

Strategies for Addressing these Shortcomings of the Relational Model

There are two main philosophies:

1. Extend the relational model to accommodate features which overcome these shortcomings.

2. Start over from scratch. It is not feasible to extend the relational model in this way.

Both approaches have been pursued over the past 20-25 years.

Extensions to the Relational Model

A number of vendors have added special features to their relational database systems.

Critical constraint:

- The extension should be compatible with the existing SQL:2003 standard.

Vendors which have followed this line with proprietary commercial products:

  - Oracle
  - IBM
  - HP
  - Informix / Illustra / Miro
  - UniSQL

- Although some features may be similar, they are to a large degree not compatible beyond the older SQL-92 level.

In addition, there have been attempts to extend SQL(-92) to accommodate desired features.

- SQL:1999 (also called SQL3, SQL-99): A standard which has recently been completed, and which addresses some of these concerns.

- SQL:2003 (also called SQL4, SQL:200n): A standard currently under development, which will address other issues.

- These standards attempt to be compatible with the earlier version of SQL (SQL2, SQL-92), with only small changes.

Fundamentally Object-Oriented Systems

During the past 15 years, a number of object-oriented database systems have been developed. These systems largely abandon the relational model, and start from scratch with an object-oriented one. Some key examples are:

- O_2
- GemStone
- ObjectStore

- Each system displayed strength for certain types of applications.

- The systems are not compatible with one another.

After this initial phase of system development, the various vendors began to develop a standard for the next generation of systems.

- The group is called ODMG (Object Database Management Group).

Overall Summary of Current Directions

1. **Bring database ideas into the existing object-oriented world:**
    - The database model is inherently object oriented; relational ideas are abandoned.
    - There is no stand-alone query language.
    - Access to the database requires a host OO programming language.

➢ Emerging standard: ODMG (Object Data Management Group) proposals.

2. **Bring object-oriented concepts into the existing relational database world:**
    - The relational model is extended to admit certain object-oriented ideas.
    - Access is via an extended version of SQL.
    - Access via queries embedded in programming languages is also possible.

➢ Emerging standards: SQL:1999, SQL:2003.

In addition, the following developments are highly relevant.

3. **Develop a general framework for manipulating objects in an interoperable environment.**
    - Framework is not specific to DBMS.
    - It deals with general object services in a distributed, heterogeneous environment.

➢ Exisiting standard: OMG (Object Management Group) proposals.

Summary of the Efficacy of These Approaches

SQL:1999/SQL:2003:

- The extensions do add some needed features. However...

- The definitions seem to be ad hoc, and not based upon sound object-oriented theory.

ODMG proposal:

- The foundations are much more solid, relative to the foundations of object-oriented systems. However...

- Many of the advantages of the relational model are lost.

  - One needs a fair amount of expertise to use such systems.
  - Schema design is a **much** more involved process.
  - In many cases, the systems are oriented towards a particular OO host language.

OMG Framework:

- The ideas embodied in this proposal are already becoming a standard.
- The details are outside of the scope of this course.

The Bottom Line

Question: Which is best?

- The relational model.
- Object-relational models
- Object-oriented models

The inescapable answer:

- It depends upon the application at hand.

- No one of these is superior to the others in all possible situations.

A better understanding of these approaches can help one to decide which is most appropriate for a given application, however.