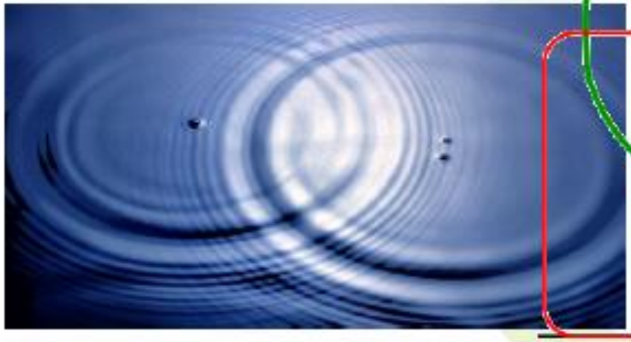
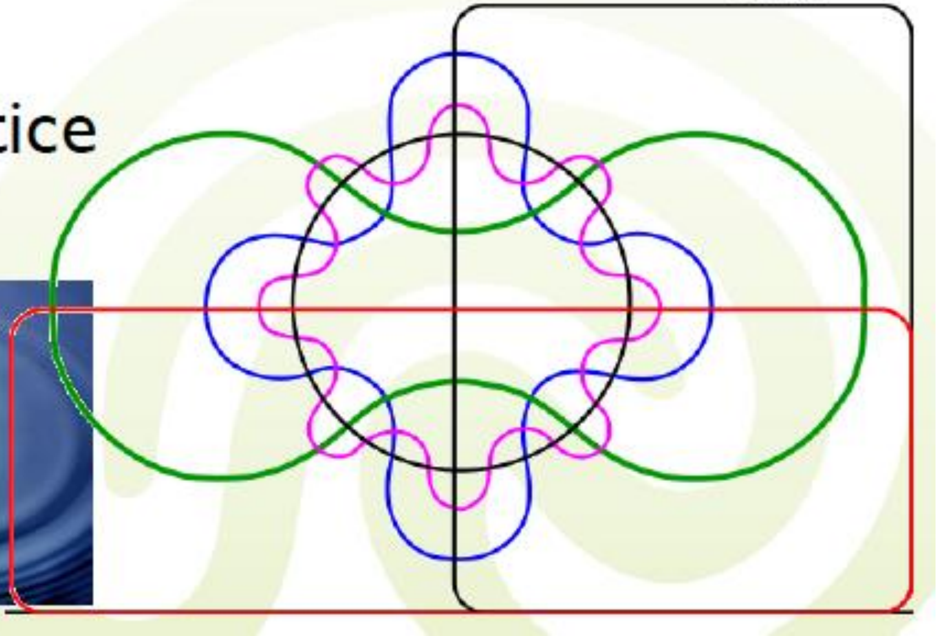
CS354: Database

Overview

- **Basic set theory**
- Relational data model
- Transformation from ER
- Integrity Constraints
- From Theory to Practice



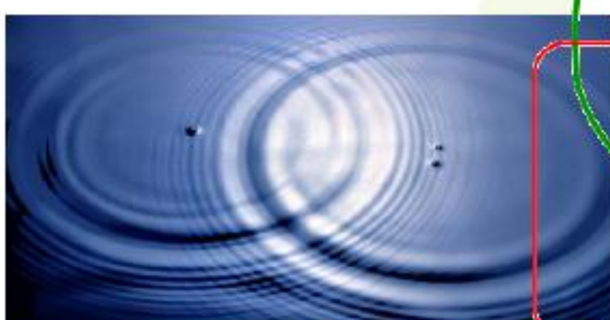


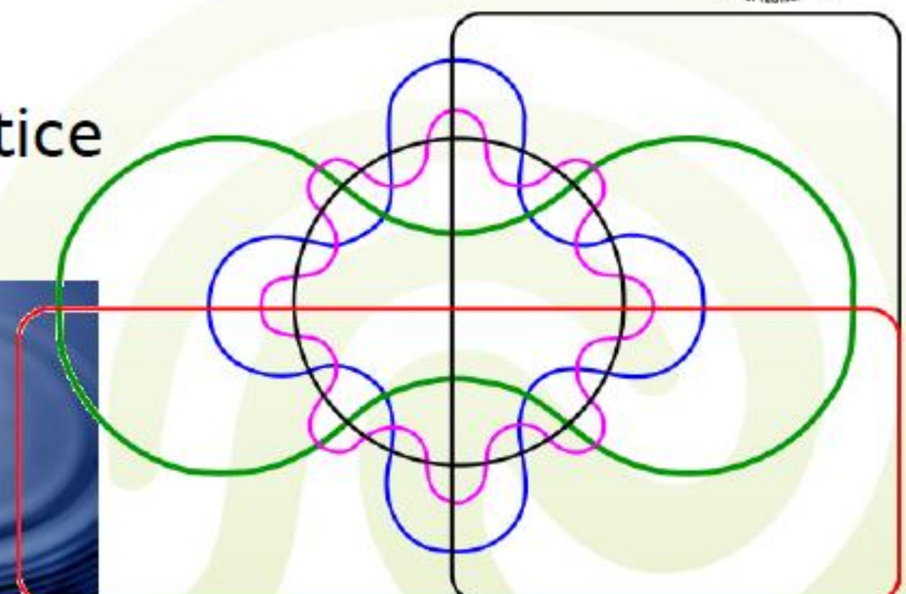
Basic Set Theory

- **Set theory** is the foundation of mathematics
  - You probably all know these things from your math course, but repeating never hurts
  - The **relational model** is based on set theory; understanding the basic math will help a lot

Overview

- Basic set theory
- **Relational data model**
- Transformation from ER
- Integrity Constraints
- From Theory to Practice





Relations

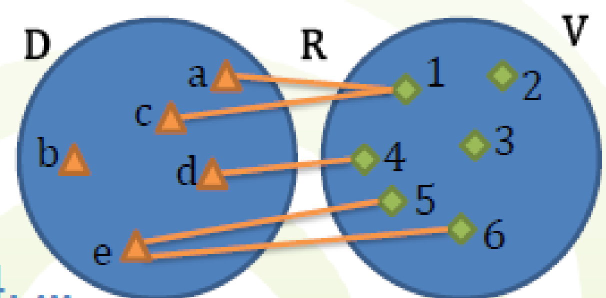
- A **relation** R over some sets $D_1, \ldots, D_n$ is a **subset** of their **Cartesian** product
  - $R \subseteq D_1 \times \ldots \times D_n$
  - The elements of a relation are **tuples**
  - The D_i are called **domains**
  - Each D_i corresponds to an **attribute** of a tuple
    - n=1: Unary relation or **property**
    - n=2: Binary relation
    - n=3: Ternary relation
    - …

Relations

- Some important properties:
  - Relations are sets in the mathematical sense, thus **no duplicate tuples** are allowed
  - The **list of tuples** is **unordered**
  - The **list of domains** is **ordered**
  - Relations can be modified by…
    - **inserting** new tuples,
    - **deleting** existing tuples, and
    - **updating** (that is, modifying) existing tuples.

Relations

- A special case: Binary relations
  - $R \subseteq D_1 \times D_2$
    - D_1 is called **domain,** D_2 is called **co-domain** (range, target)
  - Relates objects of two different sets to each other
  - R is just a set of ordered pairs
  - $R = \{\langle a,1\rangle, \langle c,1\rangle, \langle d,4\rangle, \langle e,5\rangle, \langle e,6\rangle\}$
    - Can also be written as a**R**1, c**R**1, d**R**4, ...
  - Imagine **Likes** $\subseteq$ **Person** × **Beverage**
    - Joachim Likes Coffee, Tilo Likes Tea, ...



Relations

- Example:
  - accessory = {spikes, butterfly helmet}
  - material = {silk, armor plates}
  - color = {pink, black}

color × material × accessory =
{<pink, silk, butterfly helmet>,
<pink, silk, spikes>,
<pink, armor plates, butterfly helmet>,
<pink, armor plates, spikes>,
<black, silk, butterfly helmet>,
<black, silk, spikes>,
<black, armor plates, butterfly helmet>,
<black, armor plates, spikes>}

Relations

- Relation **FamousHeroCostumes** $\subseteq$ **color × material × accessory**

FamousHeroCostumes =
{<pink, silk, butterfly helmet>,
<black, armor plates, spikes>}

Relational Model

- Well, that's all nice to know… but:
  we are here to learn about **databases!**
  - Where is the connection?
- **Here it is…**
  - A **database schema** is a description of concepts in terms of attributes and domains
  - A **database instance** is a set of objects having certain attribute values

STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

PREREQUISITE

CourseNumber	PrerequisiteNumber

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade

Schema diagram

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

A database that stores student and course information.

Relational Model

- OK, then…
  - Designing a database schema (e.g., by ER modeling) determines entities and relationships, as well as their corresponding **sets of attributes** and associated **domains**
  - The **Cartesian product** of the respective domains is the set of all possible instances (of each entity type or relationship type)
  - A **relation** formalizes the **actually existing** subset of all possible instances

Relational Model

- Database schemas are described by **relation schemas**, denoted by $R(A_1{:}D_1, \ldots, A_n{:}D_n)$
- The actual database instance is given by a set of matching **relations**
- Example
  - Relation schema:
    CATS(name : varchar(10), age : integer)
  - A matching relation:
    { (Blackie, 10), (Pussy, 5), (Fluffy, 12) }

Relational Model

- Relations can be written as **tables:**

relation name

attributes

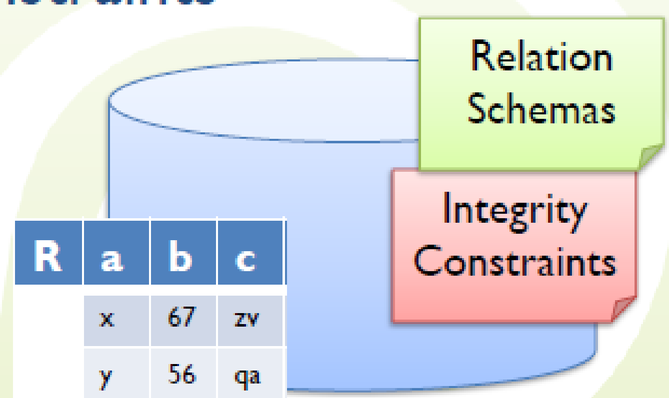
PERSON	firstName	lastName	sex
	Clark Joseph	Kent	m
	Louise	Lane	f
	Lex	Luthor	m
	Charles	Xavier	m
	Erik	Magnus	m
	Jeanne	Gray	f
...	Ororo	Munroe	f
	Tony Edward	Stark	m
	Matt	Murdock	m
	Raven	Wagner	f
	Robert Bruce	Banner	m

tuples

domain values

Relational Model

- A **relational database schema** consists of
  - a set of relation schemas
  - a set of integrity constraints
- A **relational database instance** (or state) is
  - A set of relations adhering to the respective schemas and respecting all integrity constraints



STUDENT

Name	StudentNumber	Class	Major

COURSE

CourseName	CourseNumber	CreditHours	Department

PREREQUISITE

CourseNumber	PrerequisiteNumber

SECTION

SectionIdentifier	CourseNumber	Semester	Year	Instructor

GRADE_REPORT

StudentNumber	SectionIdentifier	Grade

Schema diagram

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

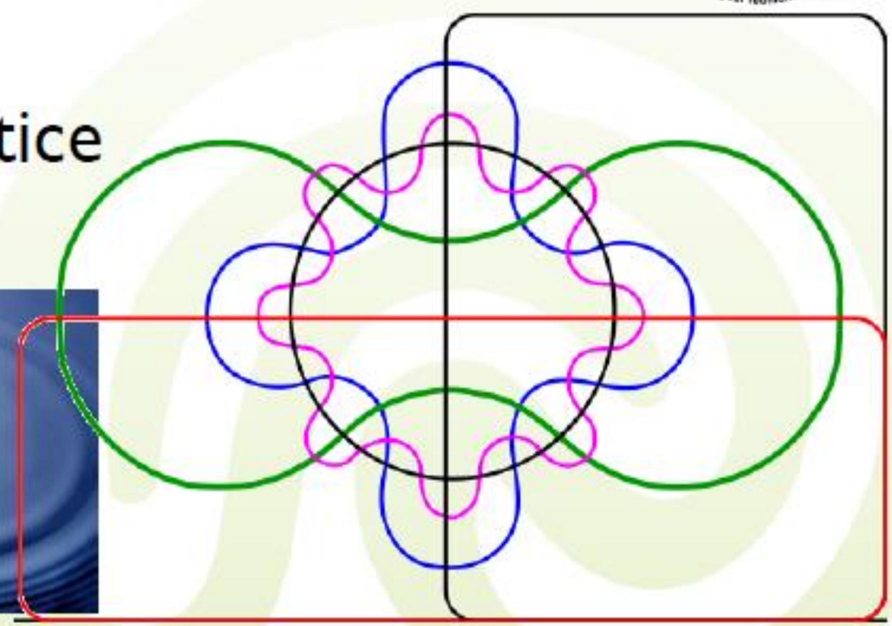
A database that stores student and course information.

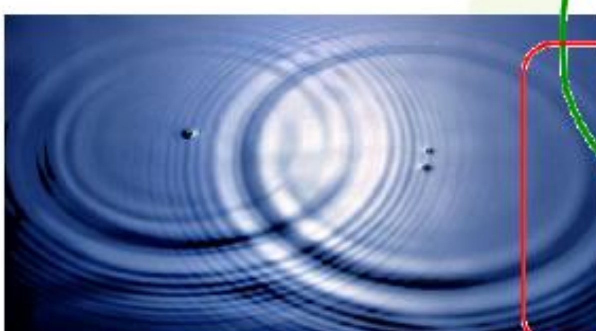
Relational Model

- Every relational DBMS needs a language to define its relation schemas (and integrity constraints)
  - **Data definition language** (DDL)
  - Typically, it is difficult to formalize all possible integrity constraints, since they tend to be complex and vague
- A relational DBMS also needs a language to handle tuples
  - **Data manipulation language** (DML)
- Today's RDBMS use **SQL** as both DDL and DML

Overview

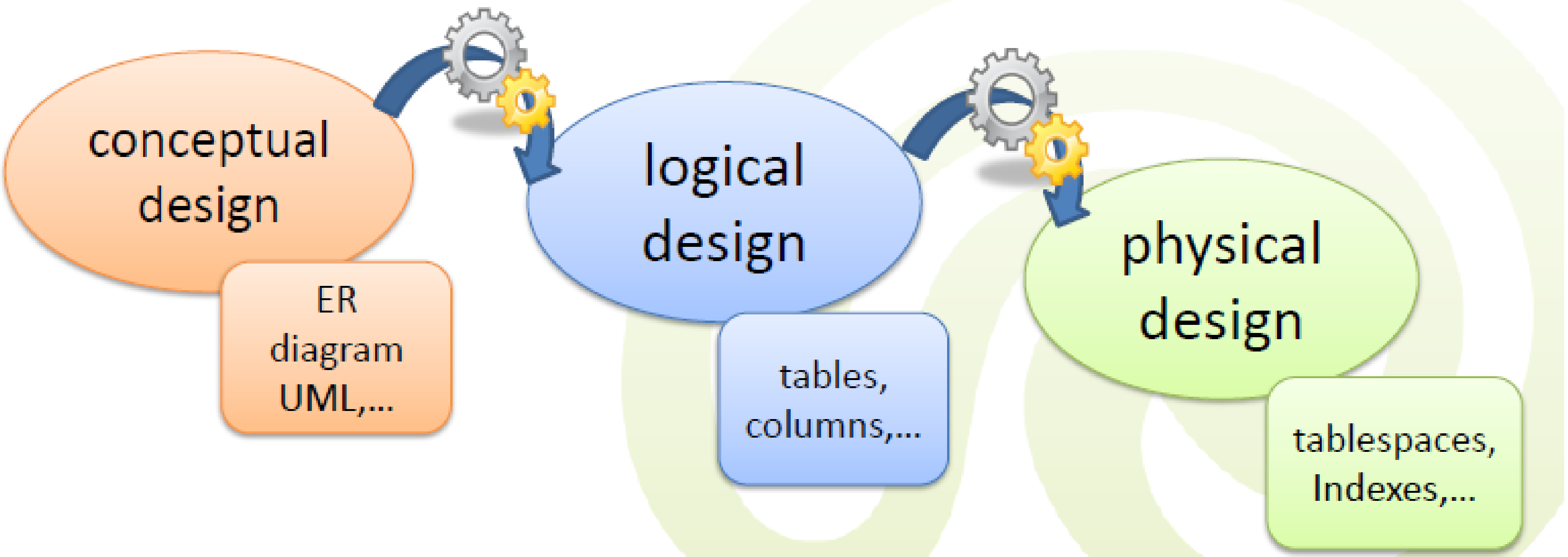
- Basic set theory
- Relational data model
- **Transformation from ER**
- Integrity Constraints
- From Theory to Practice





Conversion from ER

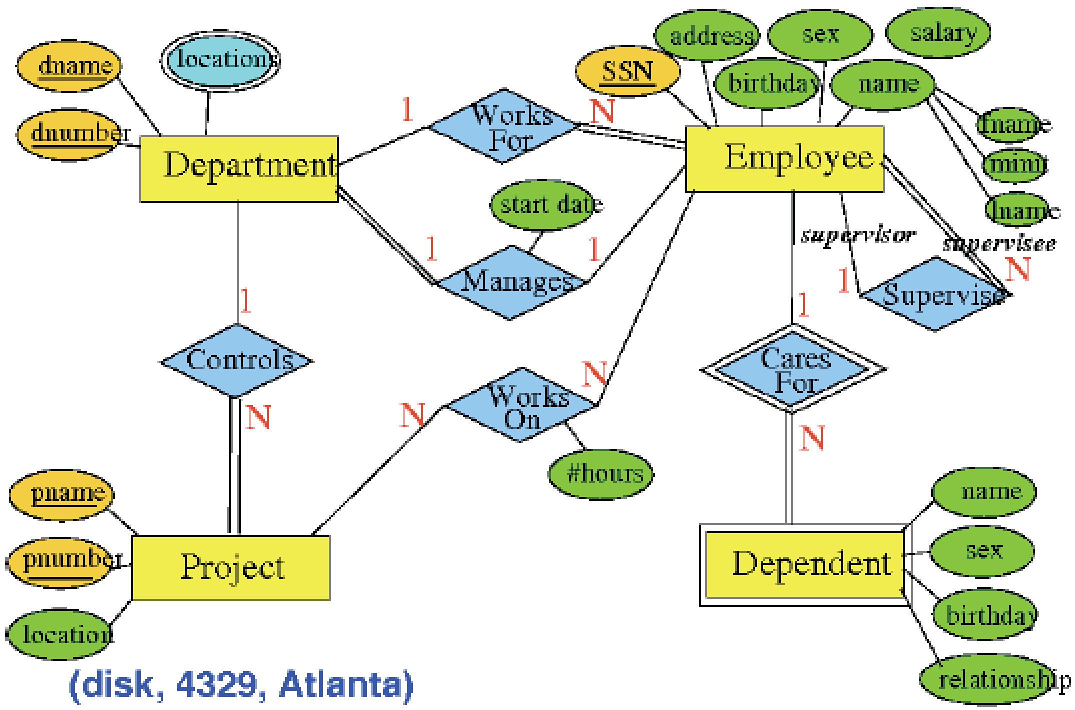
- After modeling a conceptual schema (e.g., using an **ER diagram**), the schema can be **automatically** transformed into a **relational schema**
- Remember:



ER-to-Relational Mapping

- Step 1: Convert Entities to Relations
  - Basic case: entity set E —> relation with attributes of E
  - Special case: weak entity & multi-valued attributes
- Step 2: Map Relationships to Relations
  - Basic case: relationship R —> relation with attributes being keys of related entity sets and attributes of R
  - Special case: expansion, merging, & n-ary relationship types

Example: Company database

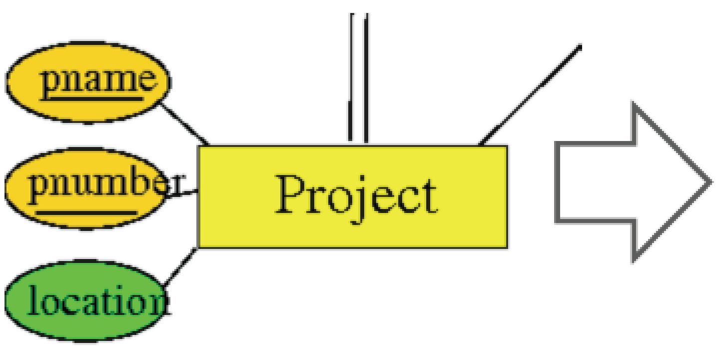


(disk, 4329, Atlanta)
(CPU, 1562, Boston)
(Printer, 7862, Denver)
...

Example from Prof Cheung's lectures

Step 1: Entity to Relation

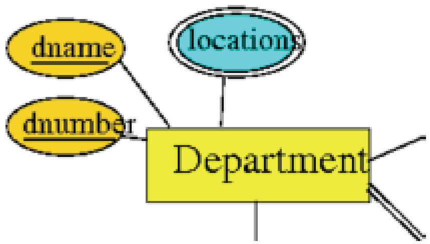
- Each entity E in the ER model is represented by one relation R in the relational model
    - Simple attributes are included in R
    - Choose a key attribute of E as a primary key for R



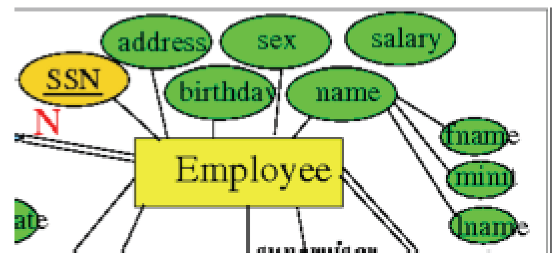
PROJECT

PName	<u>PNumber</u>	Location
...	...	...

Example: Entities to Relation



...



EMPLOYEE

SSN	FName	MI	LName	Sex	Address	BDate	Salary
...	...	...					

DEPARTMENT

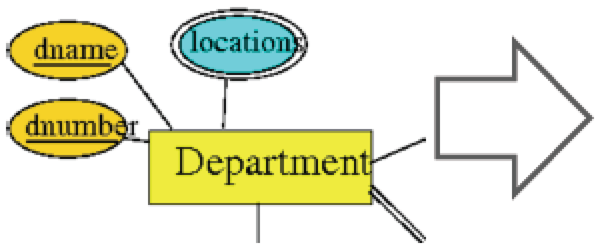
DNumber	DName	{Locations}
...	...	...

Attribute values are not ATOMIC!

Multi-valued Attribute

- Naive storing of multi-valued attributes:
  - Variable-length records causes inefficient in storage
  - Multiple tuples leads to lots of redundancy
- Use the key concept
  - Convert multi-valued attribute to new relation X
  - Add primary key to that relation

Example: Multi-valued Attribute



DEPARTMENT

DName	DNumber
Manufacturing	D1234
Research	D7652
...	...

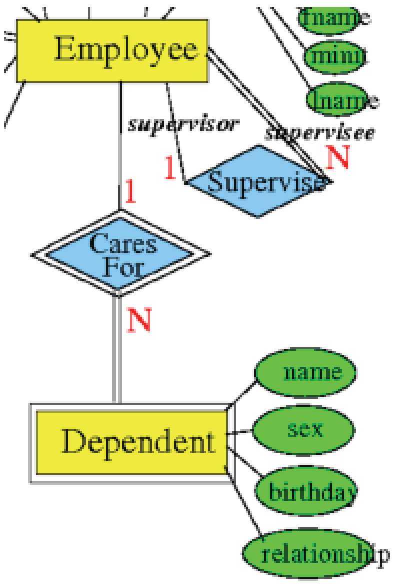
DEPARTMENT LOCATION

DNumber	Location
D1234	Atlanta
D1234	New York
D1234	Denver
D7652	San Jose
D7652	Austin
...	...

Weak Entity Type Mapping

- Weak entity does not have a key: violation of relation having a key
- Borrow key from the other entity in the identifying relationship (E) and add it to the weak entity (W)
- Result: key of weak entity consists of the key of the related entity and some identifying attribute of the weak entity

Example: Weak Entity to Relation



Necessary to identify weak entity

DEPENDENT

ESSN	FName	Sex	BDate	Relationship
	...	...	...	

Step 2: Relationship to Relation (1)

Create a new relation (S — R — T)

- New tuples of relationship R stored in this table with foreign keys from the entities S and T
- Pro: always possible
- Con: Increasing the number of relations

Step 2: Relationship to Relation (2)

Expand an existing relation (foreign key approach)

- Tuples of relationship are stored inside the table of an existing entity
- Use key of that entity to store tuples of the relationship
- Pro: only makes an existing relation a bit larger
- Con: not always possible

Step 2: Relationship to Relation (3)

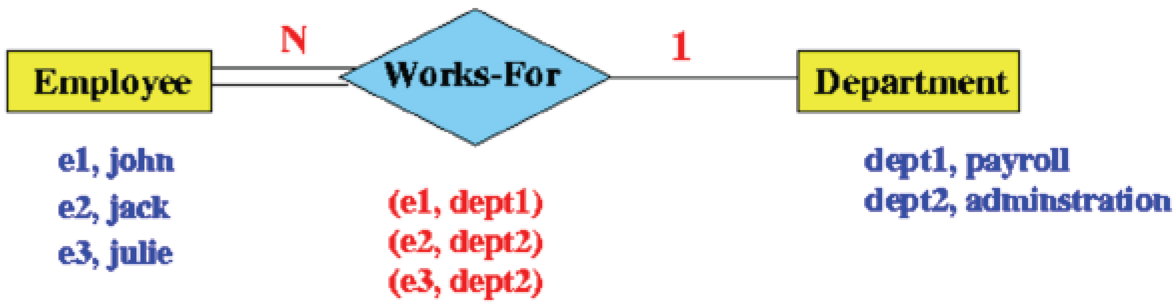
Merge two existing relations

- Merge two entity types and relationship into one relation
- Only possible in 1:1 mapping and both have total participation
- Pro: reduction of relations
- Con: rarely used

Relation Mapping Design Principles

- Relationship R where Entity1: Entity2 = 1:N —> expand the relation that represents Entity2
- Relationship R where Entity1: Entity2 = 1:1 -> expand either Entity1 or Entity2
- Avoid having attributes that can take on NULL values (e.g., expand a relationship where entity is total participation over entity with partial participation)

Example: Expansion of Works-For



EMPLOYEE

SSN	FName	MI	LName	Sex	Addres	BDate	Salary	DNo
…	…	…						

DEPARTMENT

DName	DNumber
…	…

Example: Expansion

PROJECT

Controls-Project
Dept:Project = 1:N

PName	<u>PNumber</u>	Location	DNum
...	...	...	

Supervisor
Supervisor:Supervisee = 1:N

EMPLOYEE

<u>SSN</u>	FName	MI	LName	Sex	Address	BDate	Salary	superSSN	DNo
...	...	...							

DEPARTMENT

Manager
Employee:Dept = 1:1

DName	<u>DNumber</u>	mgrSSN	mgrStart
...	...		

Example: Creation of Works-On Relation

Why expansion doesn't work:

Employee:Project = M:N

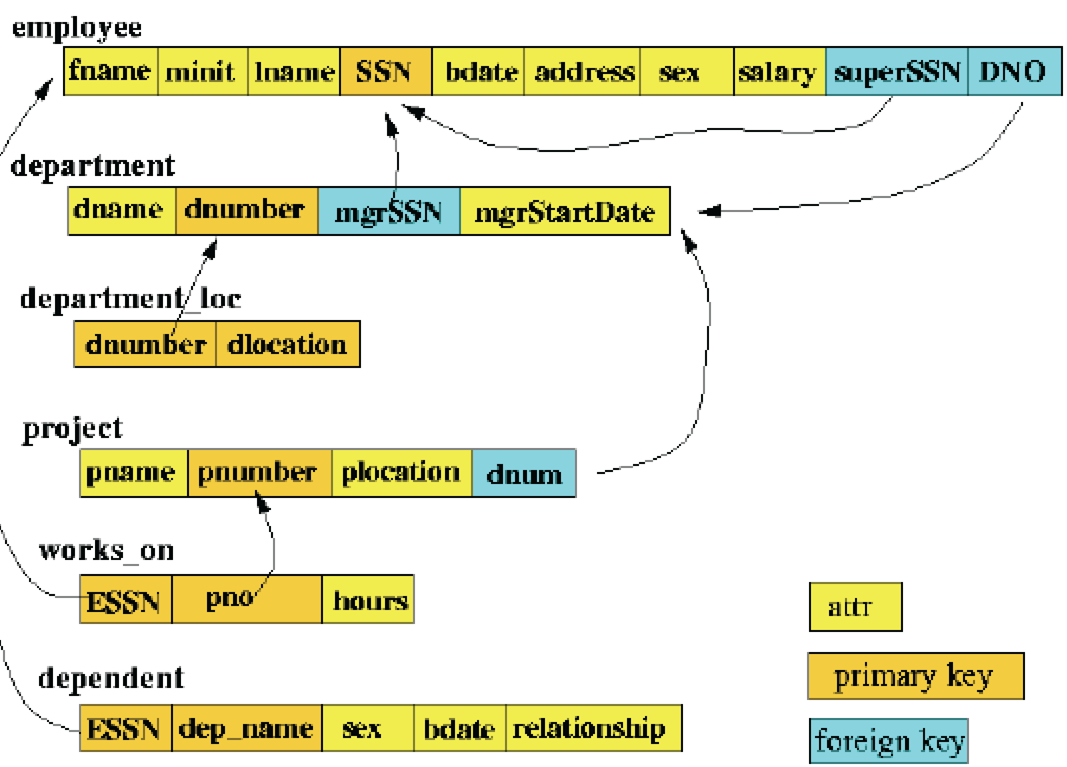
Employee:Project participation = Partial:Partial

- Expand Employee with attribute WorkedOnProject leads to multi-valued attribute
- Expand Project relation with attribute WorkerSSN also results in multi-valued attribute

WORKS_ON

ESSN	PNO	Hours
...	...	

Example: Full Relational Model



Mapping Summary

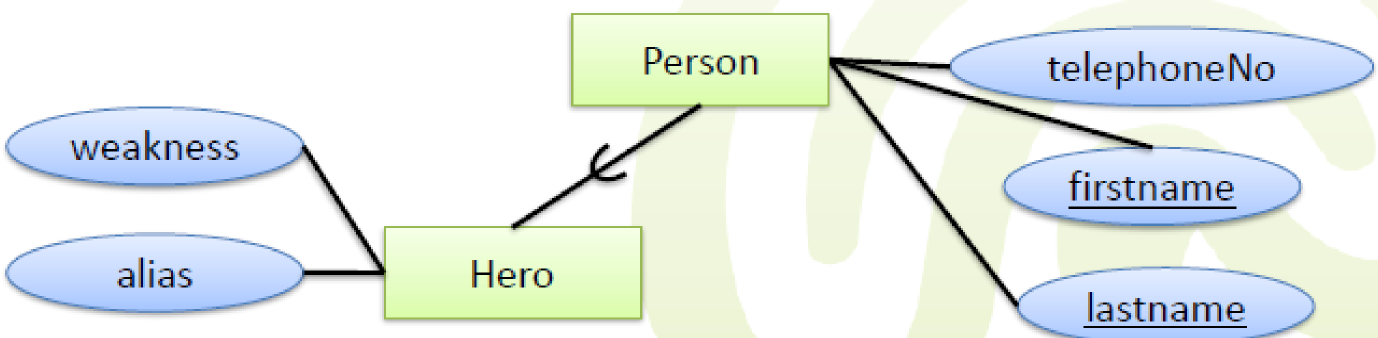
ER Model	Relational model
Entity type	Entity relation
1:1 or 1:N relationship	Expand (or create R relation)
M:N relationship	Create R relation with two foreign keys
n-ary relationship type	Create R relation with n foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Key attribute	Primary (or secondary) key

Relational Model: Recap

- Relational Model
  - Relation, attributes
  - Schema vs instance
  - Relational model constraints
- ER to Relational
  - Entity set, relationship —> relation

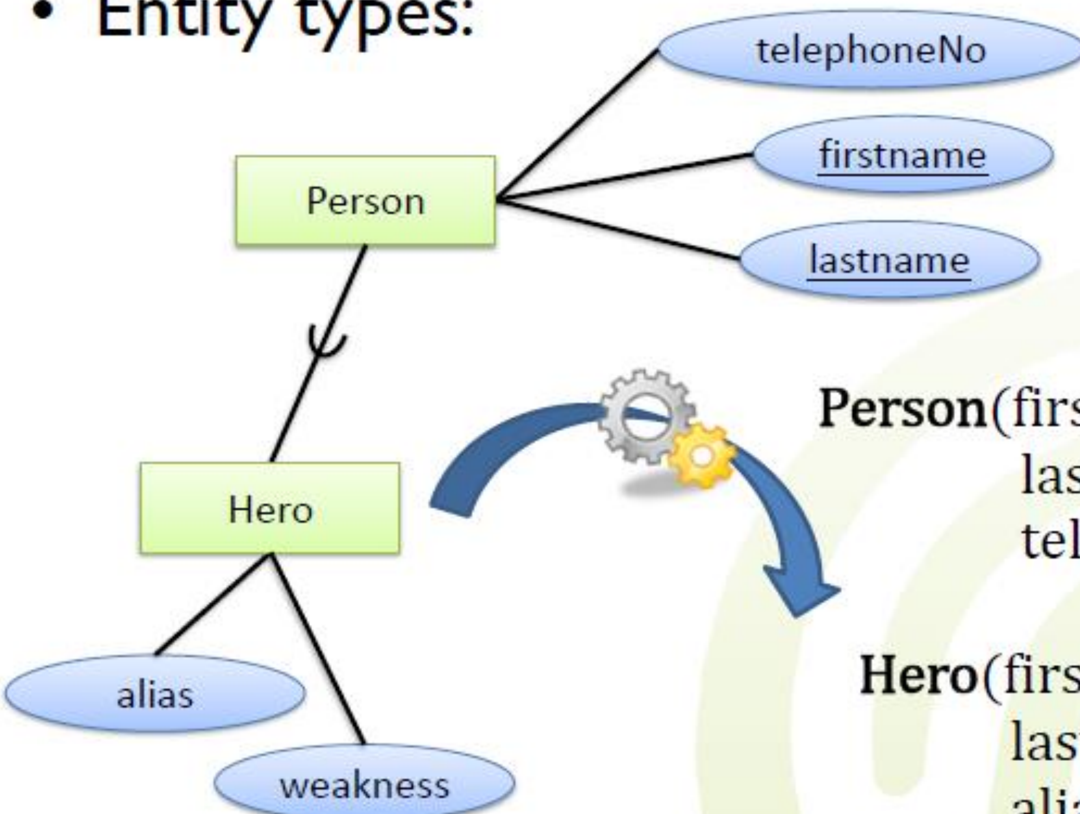
Conversion from ER

- Each **entity type** E with attributes $A_1, \ldots, A_n$ from domains $D_1, \ldots, D_n$ is converted into an ***n*-ary relation schema** $E(A_1{:}D_1,\ldots, A_n{:}D_n)$
- If there is a relationship type E is_a F involved (**specialization**), the inheritance relationship can be expressed by copying all key attributes from F



Conversion from ER

- Entity types:



Person(firstname :Varchar(20),
lastname : Varchar(30),
telephoneNo : Integer)

Hero(firstname : Varchar(20),
lastname : Varchar(30),
alias : Varchar(20),
weakness : Varchar(10))