Relational Model

CS 377: Database Systems

Relational Model

- First formal database model
- Introduced by Ted Codd in 1970
- Conceptual basis of relational databases
  - Simple and based on the mathematical relations
  - Declarative method for specifying data and queries
- Previous models include hierarchical and network models

Relation

Data is stored in **tables** (**relations**)

- Tuple is a row in the table
- Attribute is a column header in the table

attribute

PRODUCT ← table name

Name	Category	Price	Manufacter
iPad	Tablet	$399.00	Apple
Surface	Tablet	$299.00	Microsoft
Kindle	eReader	$79.00	Amazon
…	…	…	…

record/tuple

Relation Definitions

- **Domain**: set of atomic values that are assigned to an attribute (e.g., name: string, category: string, price: real)
- **Relation Schema** $R(A_1, A_2, \ldots, A_n)$: made of of a relation name R and a set of attributes $A_1, A_2, \ldots, A_n$
(e.g., Product(name, category, price, manufacturer)
  - In practice, the domain is added for each attribute
- **Degree of a relation**: number of attributes in the relation schema
  - this is different than the degree in ER model!

Schema and Instances

- **Database schema**: a collection of relation schemas
- **Instance of a relation**: set of tuples or records
- **Instance of a database**: a collection of relation instances
- Can view schemas as types while instances as values in a programming language
- Schemas are stable over long periods of time while instance changes constantly with data inserts, updates, and deletions

Relational Model Notation

Notation	Description
$R(A_1, A_2, \ldots, A_n)$	Relation schema R of degree n
Q, R, S	Relation names
q, r, s	Relations
t, u, v	Tuples
$t(a_1, a_2, \ldots, a_n)$	tuple t of a relation
$t[A_i]$	the value of the attribute A_i in the tuple t
$t[A_i, A_j, A_k]$	value of the attributes A_i, A_j ,A_k in the tuple t

Relational Model Constraints

- Restrictions on actual values in a database
- **Inherent model-based constraints** or **implicit constraints**: inherent in the data model (e.g., no duplicate tuples)
- **Schema-based constraints** or **explicit constraints**: can be directly expressed in schemas of the data model
- **Application-based / semantic constraints**, or **business rules**: cannot be directly expressed in schemas and can only be enforced and expressed in the application program

Schema-based Constraints: Domain Constraints

- Value of each attribute A must be an atomic value from the domain of A
- Typical data types associated with domains
  - Numeric data types for integers and real numbers
  - Characters, fixed-length or variable-length strings
  - Booleans
  - Date, Time, Timestamp
  - …

Schema-based Constraints: Key Constraints

- No two tuples can have the same combination of values for all their attributes
- **Superkey**: set of attributes in a relation R such that no 2 different tuples will have the same values for that set of attributes

$$\forall t_1, t_2 \in R : t_1[SK] \neq t_2[SK]$$

- **Key**: minimal set of attributes in relation R such that no 2 tuples have the same values (i.e., key is a minimal superkey)
- **Candidate key**: any key

Schema-based Constraints: Key Constraints (2)

- **Primary key**: key chosen to be used to identify tuples in a relation
  - Once chosen, you must use that primary key throughout the database
  - Other candidate keys are unique keys
  - Every relation schema must have a primary key
- **Foreign key**: set of attributes inside some relation *R1* that is a primary key of another relation *R2*

Example: Keys

primary key → PID; candidate key → PID, SSN

PERSON

PID	SSN	Name	Address
52032	111-12-2345	John Doe	123 My Street
12345	444-23-1234	Jane Smith	555 South Street
79823	555-67-8910	Tom Thumb	224 First Street
...	...	...	...

primary key → TID; foreign key → PID

PURCHASE

TID	PID	Product	Price
123456778	52032	iPad Air 2	$399.00
123470901	52032	Kindle	$79.00
234096701	79823	Surface	$499.00
...	...	...	...

Schema-based Constraints: Entity Integrity

- The attribute values of the primary key cannot have NULL values
- Primary key is used to identify a tuple
- NULL value means not applicable or not available which hinders the ability to identify a tuple

PERSON

PID	SSN	Name	Address
52032	111-12-2345	John Doe	123 My Street
NULL	444-23-1234	Jane Smith	555 South Street
…	…	…	…

Not allowed!

Schema-based Constraints: Referential Integrity

- A tuple in one relation (t_1 in R_1) that refers to another relation (t_2 in R_2) must refer to an existing tuple in that relation (t_2 must exist)

$$t_1[FK] = t_2[PK]$$

- R_1 is the referencing relation and R_2 is the referenced relation

tuple must exist in PERSON table

PURCHASE

TID	PID	Product	Price
123456778	52032	iPad Air 2	$399.00
123470901	52032	Kindle	$79.00
234096701	79823	Surface	$499.00
...	...	...	...

ER Model vs Relational Model

ER model (conceptual model)

- Several concepts: entities, relationships, attributes
- Well-suited for capturing application requirements
- Not well-suited for computer implementation

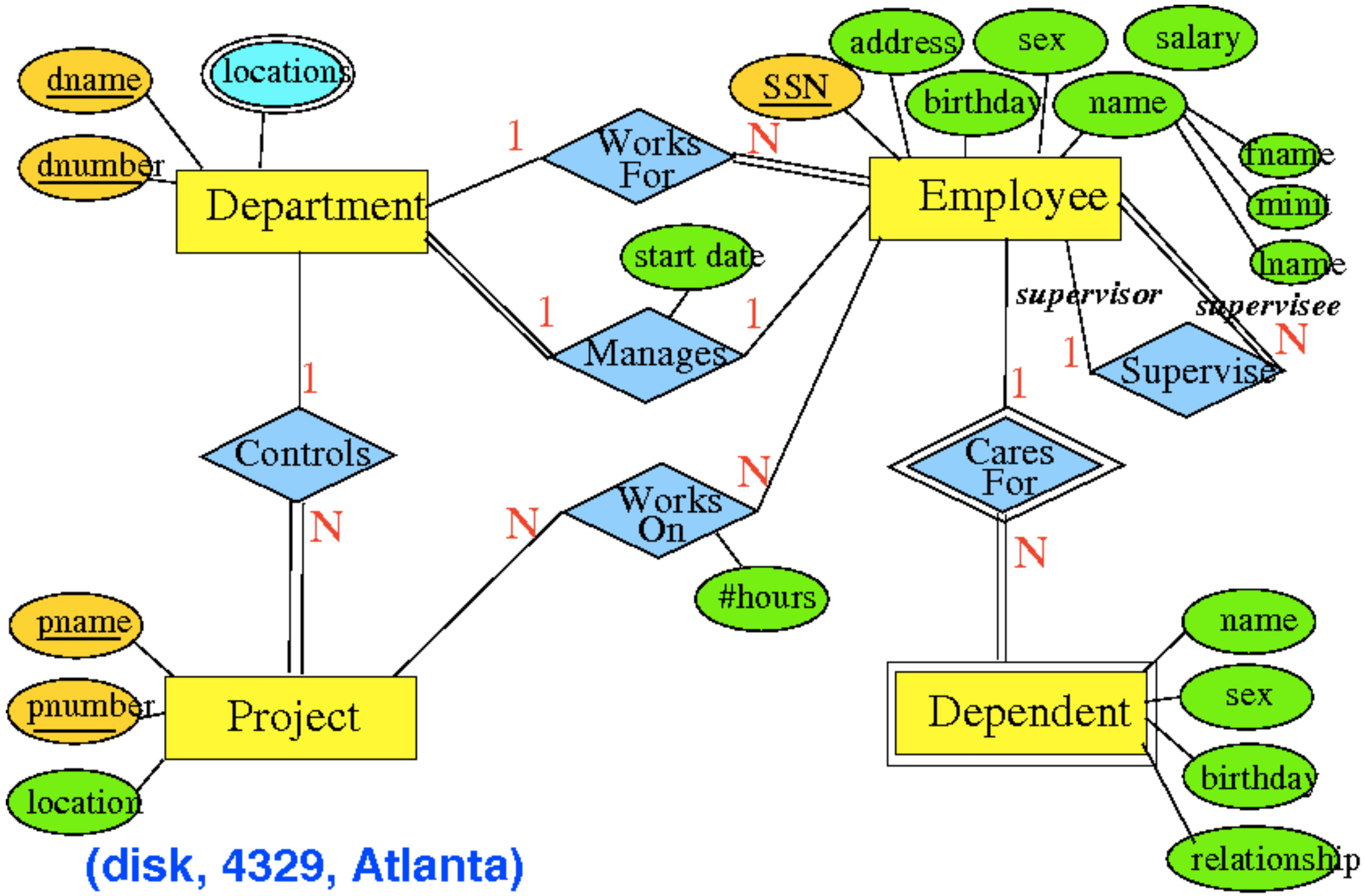
Relational model (implementation model)

- Single concept: relation (not same as mathematical concept!)
- Everything is represented with a collection of tables
- Well-suited for efficient manipulations on computers

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

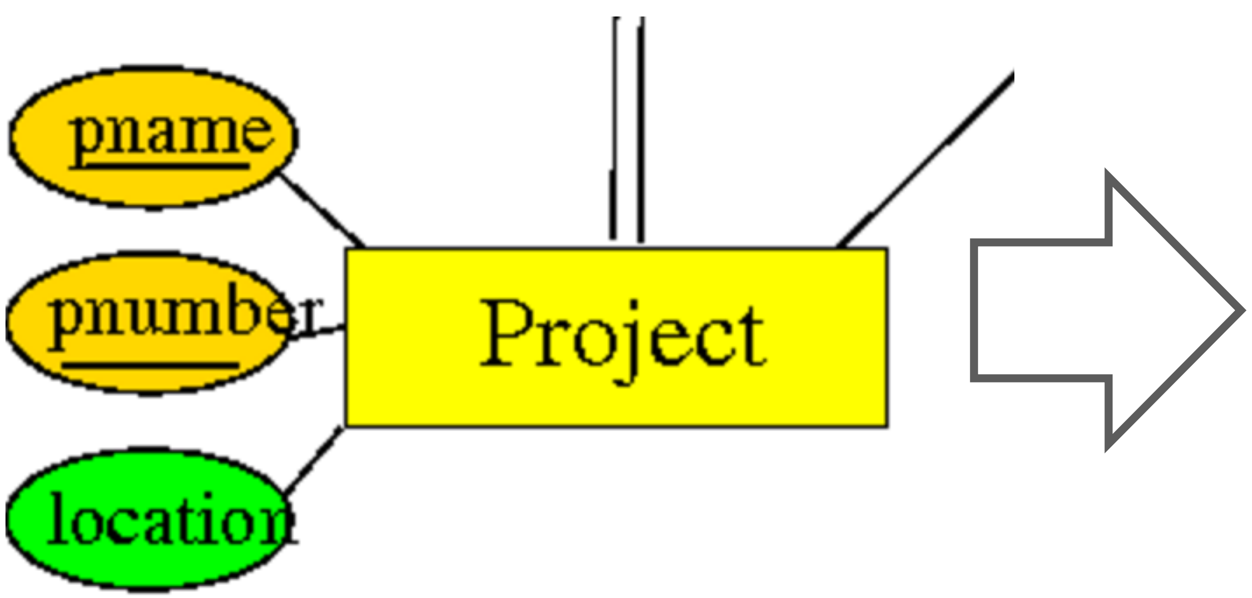


(disk, 4329, Atlanta)
(CPU, 1562, Boston)
(Printer, 7862, Denver)
...

Example from Prof Cheung's lectures

Step 1: Entity to Relation

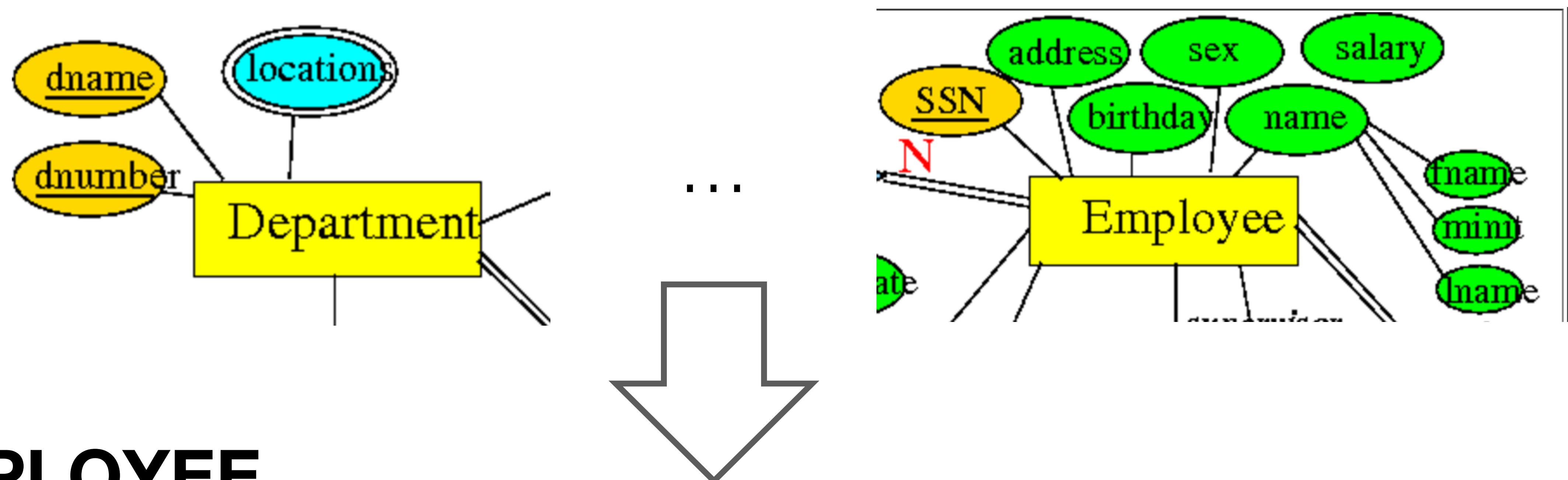
- Each entity E in the ER model is represented by one relation R in the relational model
  - Simple attributes are included in R
  - Choose a key attribute of E as a primary key for R



PROJECT

PName	<u>PNumber</u>	Location
…	…	…

Example: Entities to Relation



EMPLOYEE

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

DEPARTMENT

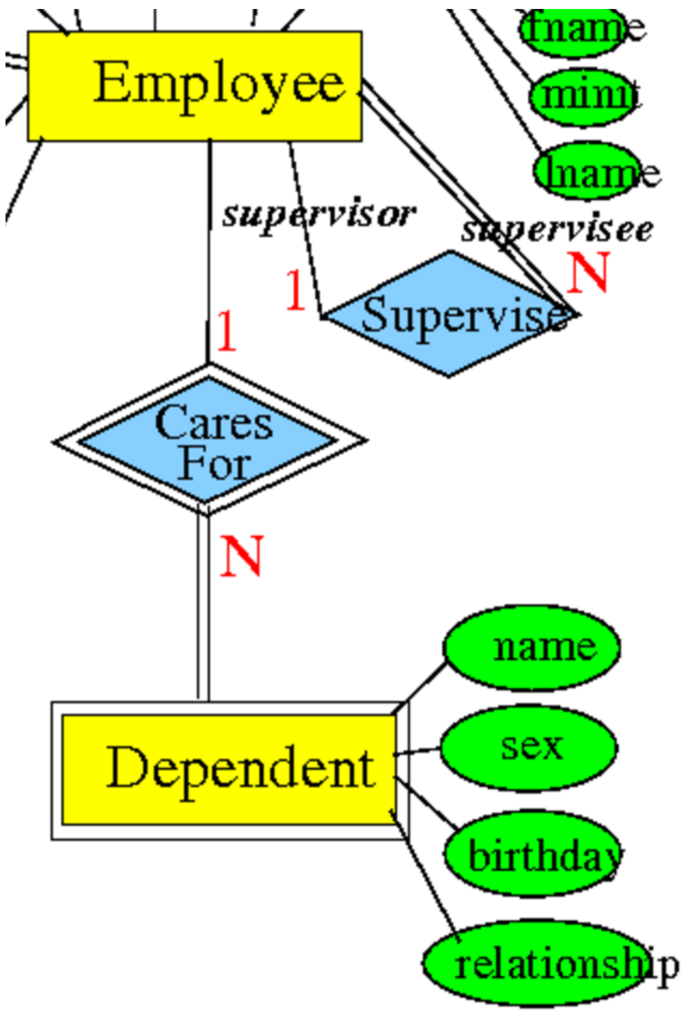
DNumber	DName	{Locations}
...	...	...

Attribute values are not ATOMIC!

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

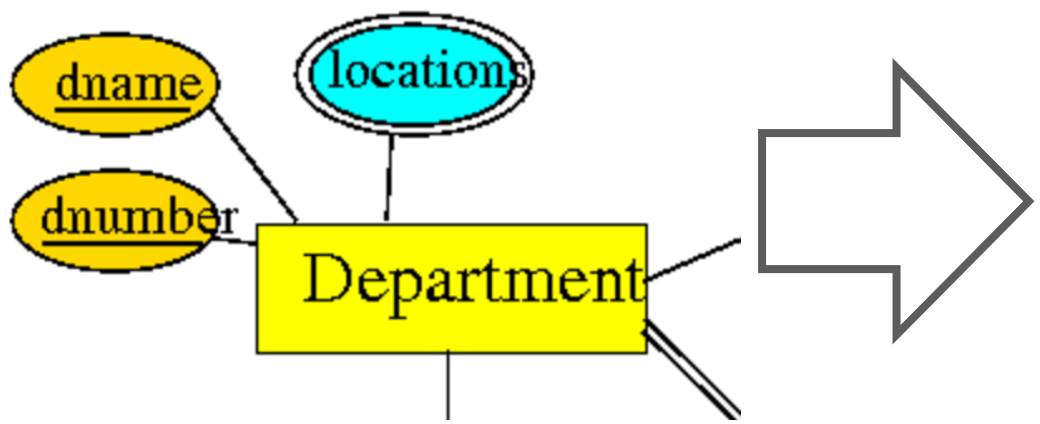
DEPENDENT

ESSN	FName	Sex	BDate	Relationship
	…	…	…	

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
…	…

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)

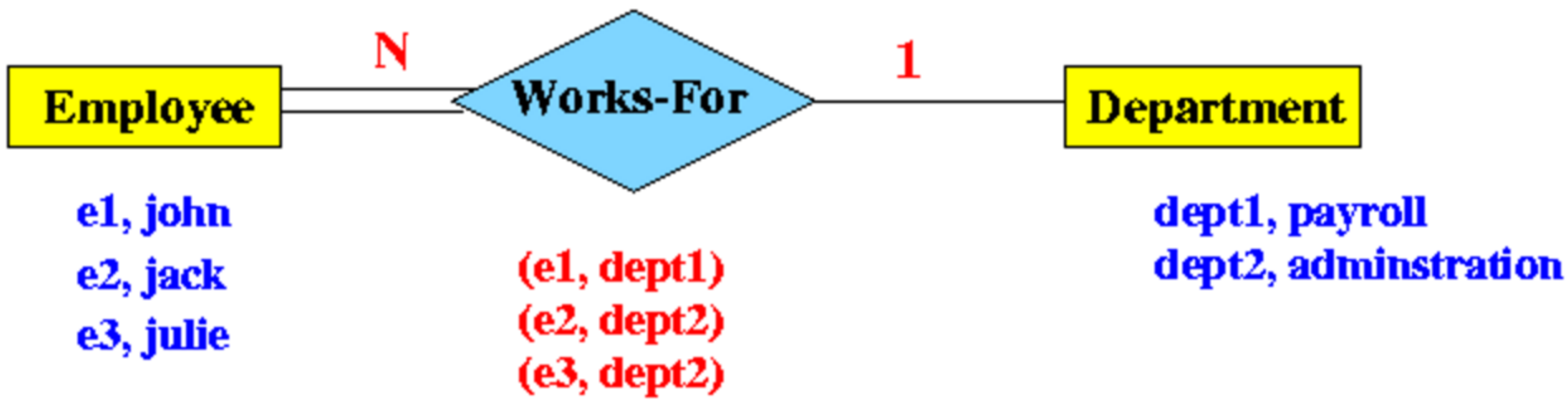
Merge two existing relations

- 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

<u>SSN</u>	FName	MI	LName	Sex	Addres	BDate	Salary	DNo
…	…	…						

DEPARTMENT

DName	<u>DNumber</u>
…	…

Example: Expansion

Controls-Project
Dept:Project = 1:N

PROJECT

PName	PNumber	Location	DNum
…	…	…	

Supervisor
Supervisor:Supervisee = 1:N

EMPLOYEE

SSN	FName	MI	LName	Sex	Address	BDate	Salary	superSSN	DNo
…	…	…							

Manager
Employee:Dept = 1:1

DEPARTMENT

DName	DNumber	mgrSSN	mgrStart
…	…		

Example: Creation of Works-On Relation

Why expansion doesn't work:

Employee:Project = M:N

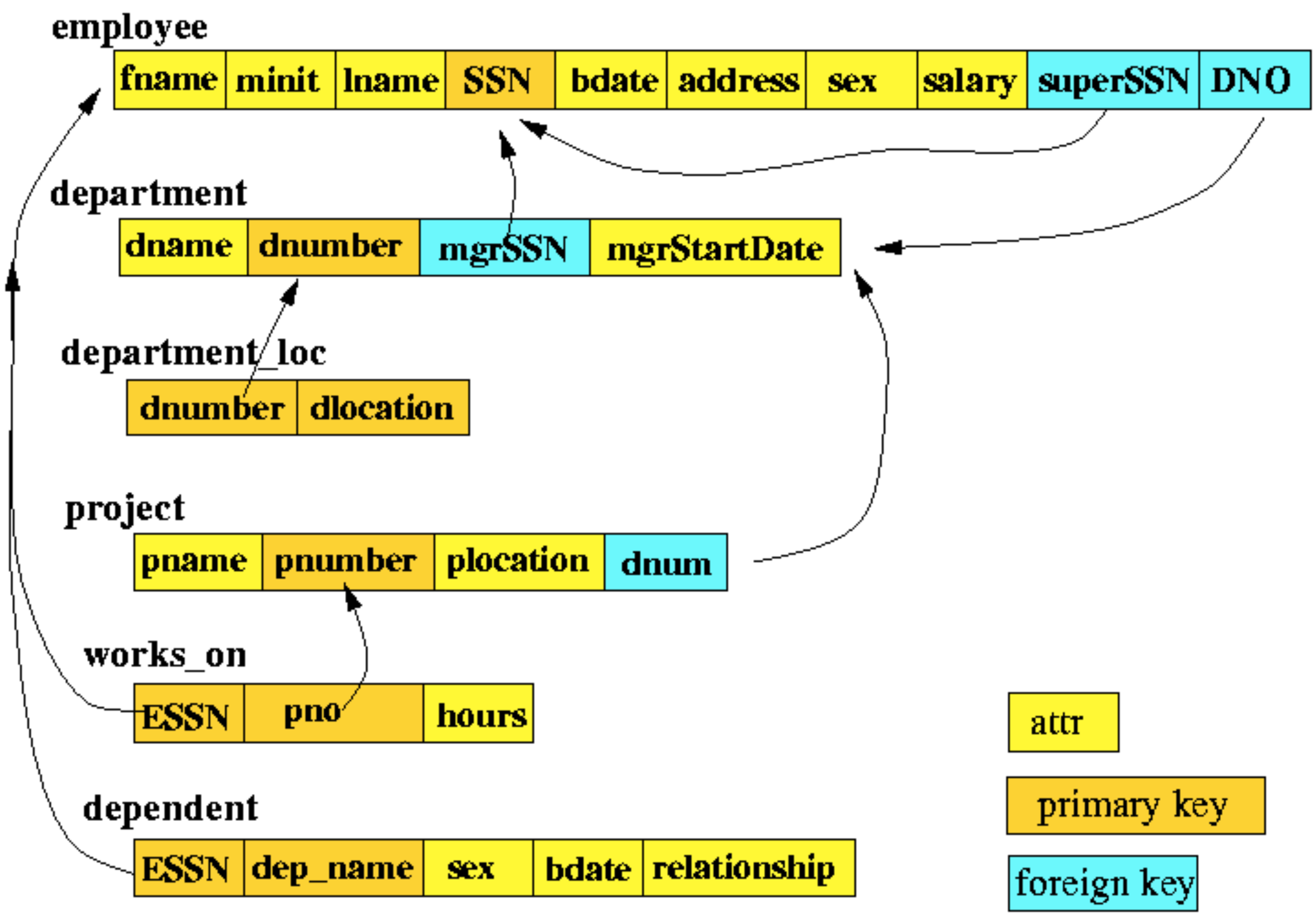
Employee:Project participation = Partial:Partial

- 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



http://www.mathcs.emory.edu/~cheung/Courses/377/Syllabus/3-Relation/rel-db-design2.html

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