DB Fundamentals, ERD

7 November 2011
Lecture 2

Topics for Today

- Intro: Queries, Transactions, Crashes
- Structure of a DBMS

- Entity Relationship Diagrams
  - Simple to Complex
  - Entities, Attributes, Entity Sets
  - Relationships and Relationship Sets
  - Additional Features

- Sources:
  - Ramakrishnan and Gehrke 1.6 – 1.9, 2.1 – 2.4

Queries

- The power of database is how it lets the user **ask questions** about its contents

- Queries are questions we ask the DBMS – it answers them by looking at the database
  - SQL is the most popular **Query Language**

- It’s important to understand the meaning of queries - their *semantics*
  - To help, abstract query languages have been devised – Relational Algebra underlies SQL

Two Types of Queries

- Read Queries
  - Ask the DBMS to answer a question by returning a single value or a table of results
  - Using the <u>Query Language</u>

- Write Queries
  - Ask the DBMS to add/delete/modify a record
  - Using the <u>Data Manipulation Language</u>

- SQL is both

Three faces of SQL

- SQL is multipurpose:
  - We can use it to define, modify, and delete relations (Data Definition Language)
  - We can use it add, modify, and delete records (Data Manipulation Language)
  - We can use it to ask questions about the data (Query Language)

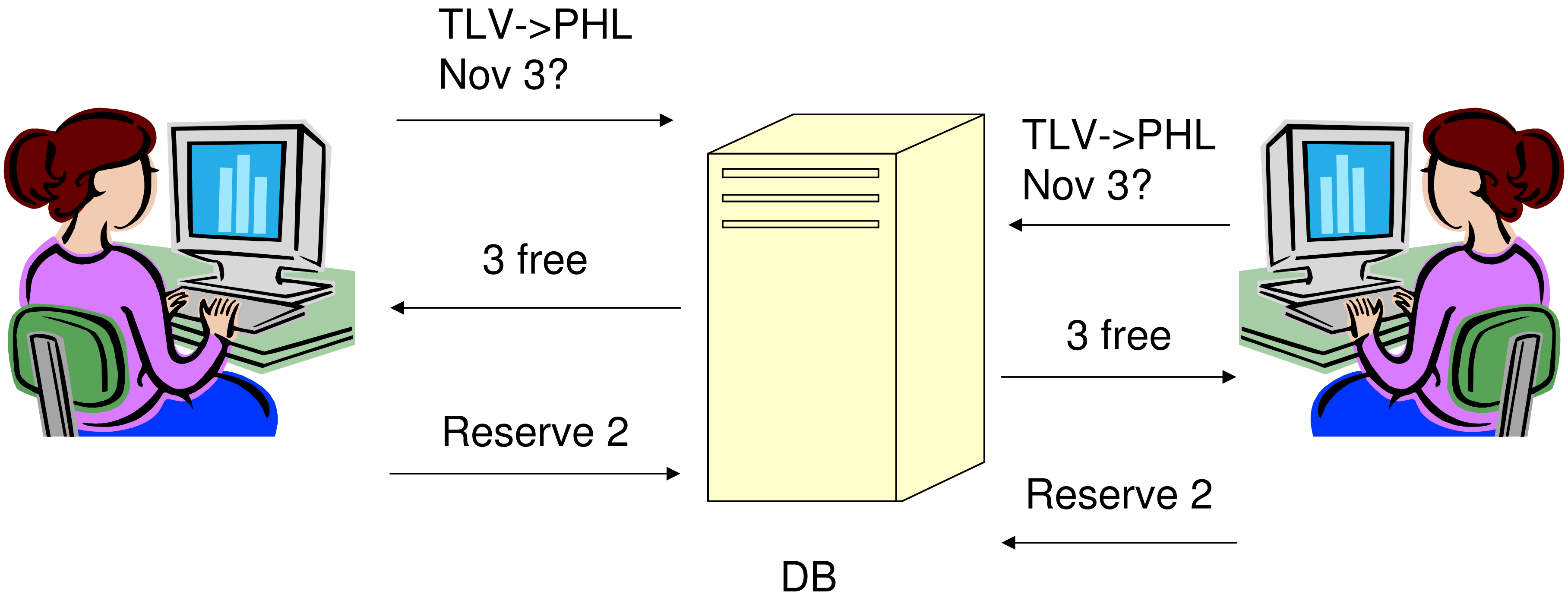
How to Query

- Could log into the DBMS with a command line or GUI interface and ask queries directly
- Could use a remote DB tool which builds a connection to a DB and has a GUI to receive user queries and display results
- Could write application code which builds a database connection, sends queries, and does things with the results
- Could build a web site which has a backend application which runs queries implicitly or explicitly and shows the results on the page

Transaction Management

- Transactions essential for two reasons:
  - Concurrency (multiple users)
  - Failure recovery

Concurrency



TLV->PHL
Nov 3?

3 free

Reserve 2

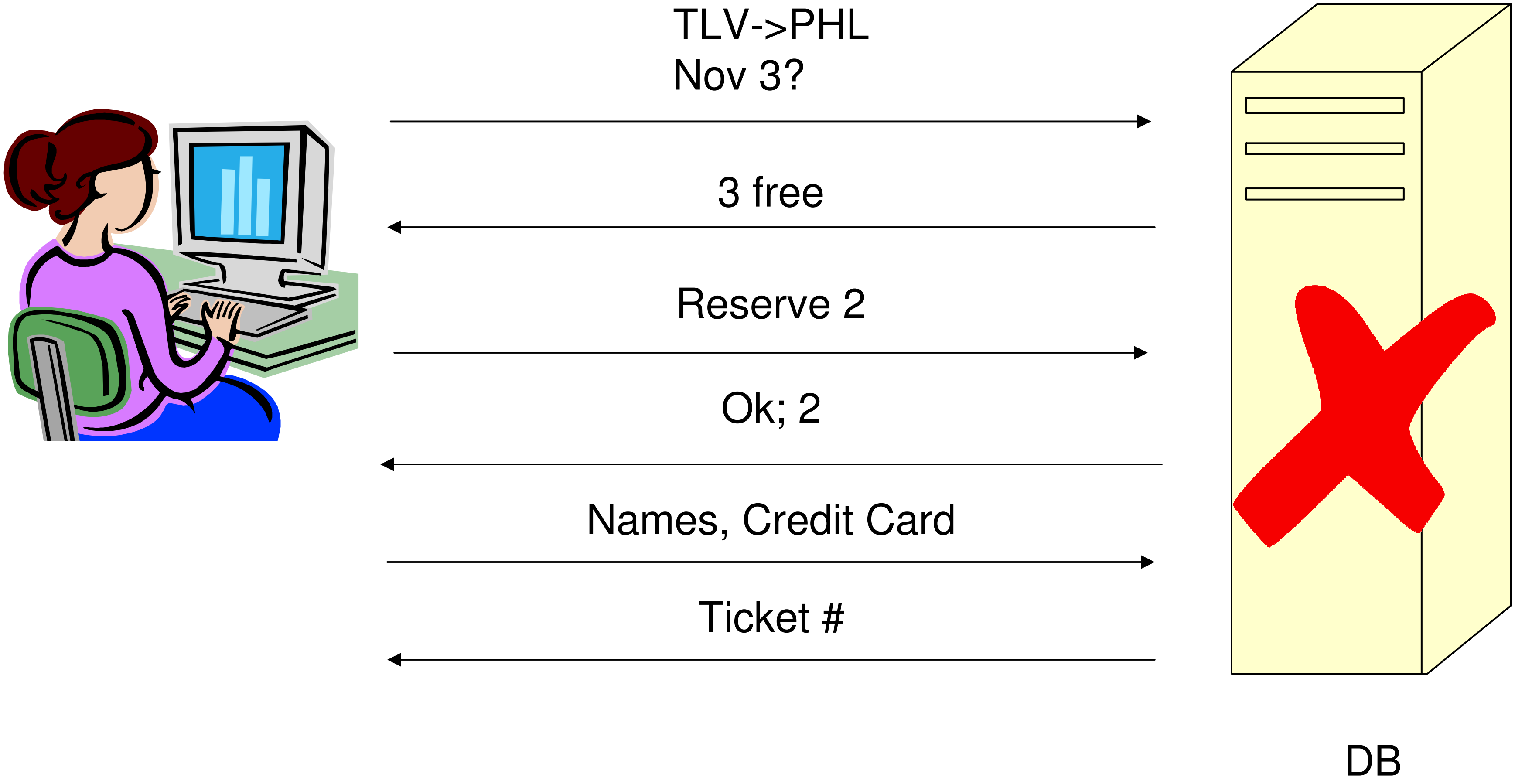
TLV->PHL
Nov 3?

3 free

Reserve 2

DB

Failure Recovery



TLV->PHL
Nov 3?

3 free

Reserve 2

Ok; 2

Names, Credit Card

Ticket #

DB

Transactions in DBs

- A transaction is one execution of a user program on a DBMS
  - Similar to the Operating System and Distributed Systems definitions
  - Unlike the definition we'll see in the other course
- DBMS performs actions at the transaction level
  - Atomic
  - All or nothing

Tools for Concurrency Control

- DBMS must manage multiple transactions
  - Enforce non-conflict
- Most use **locks**
  - There are other ways to do it, though

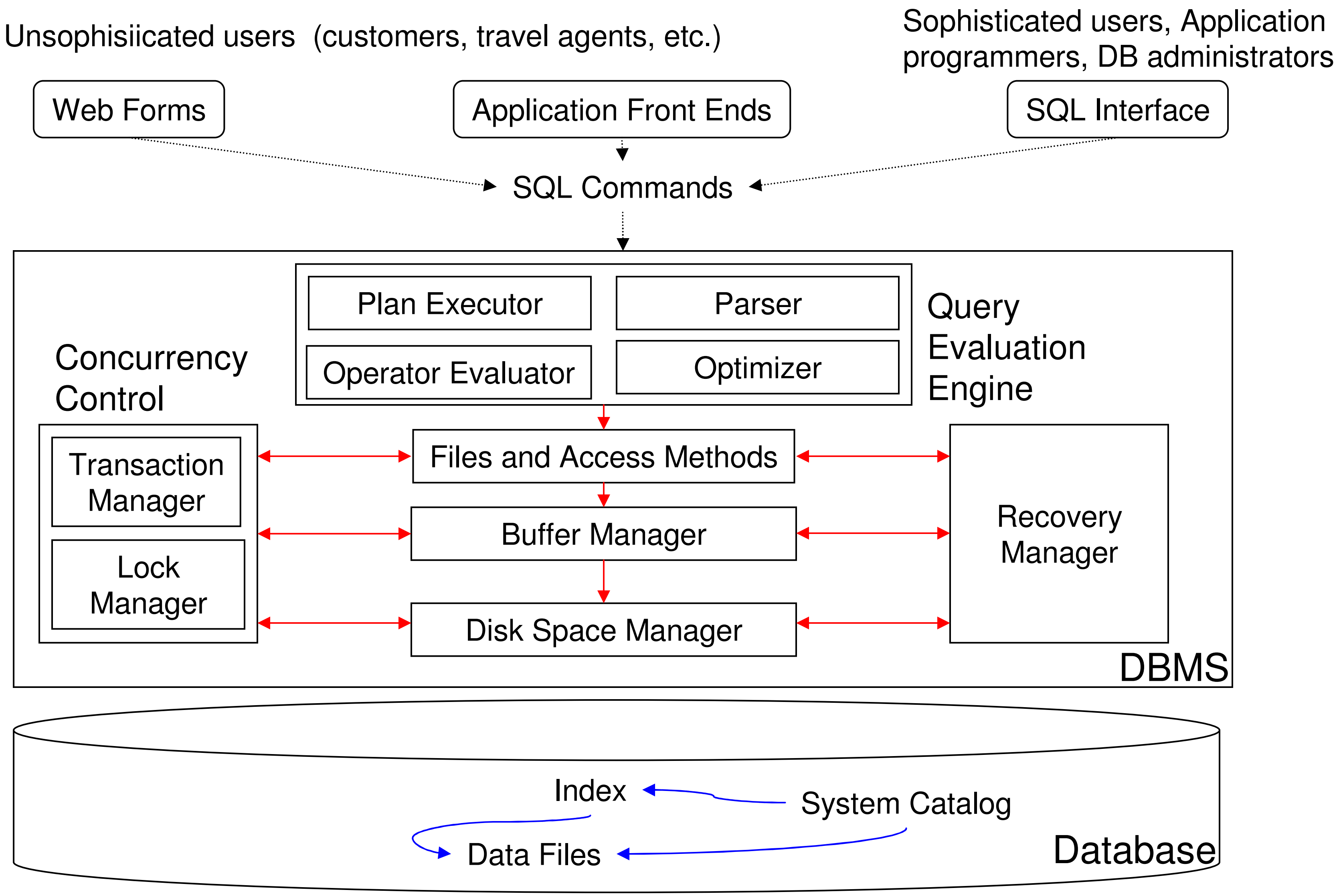
Dealing with Crashes

- The main tool to deal with crashes: the **log**
  - Write ahead – each operation is entered in the log before it is executed
  - Log is flushed every so often, always when a transaction completes
- Also:
  - Checkpoints
  - Ability to selectively force pages to disk
- When a crash occurs:
  - Reload the database to where it was before
  - Go over the log and remove actions from transactions which didn’t commit
- More in the book on this – we may get to it this semester

So Far

- Intro: Queries, Transactions, Crashes
- Structure of a DBMS
- Entity Relationship Diagrams
  - Simple to Complex
  - Entities, Attributes, Entity Sets
  - Relationships and Relationship Sets
  - Additional Features

Structure of a DBMS



So Far

- Intro: Queries, Transactions, Crashes
- Structure of a DBMS

- Entity Relationship Diagrams
  - Simple to Complex
  - Entities, Attributes, Entity Sets
  - Relationships and Relationship Sets
  - Additional Features

Why use ERD?

- Useful for:
- Requirements Analysis
  - Think about relationships and objects
  - Concretize ideas
- Conceptual Database Design:
  - Organizing data in a schema is hard – pictorial help
  - Easy to see connections and constraints
- Logical Database Design:
  - To communicate requirements between teams or designers
  - Acts as documentation

What is an Entity?

- Basically any noun you can describe:
  - Student
  - Car
  - Book
  - College
  - Courage ✗
  - Four ✗

- Attributes:
  - Descriptors of entities
  - All the ones that are interesting enough for the database

- Entities Sets are like classes – *entities* are instances

Entity Example

- A student might be identified by (attributes):
  - Student Id number
  - Name
  - National ID number
  - Address
  - Height
  - Degree track
  - Phone Number
  - Email address
  - Passport Number

- Which would be appropriate for a college student relation?

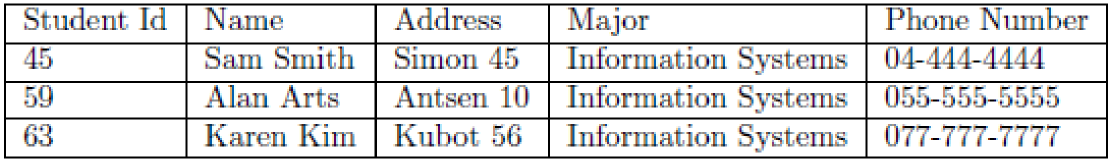
Attributes

- Attributes have a given domain
  - Strings
  - Numbers
  - Sets
  - Formats

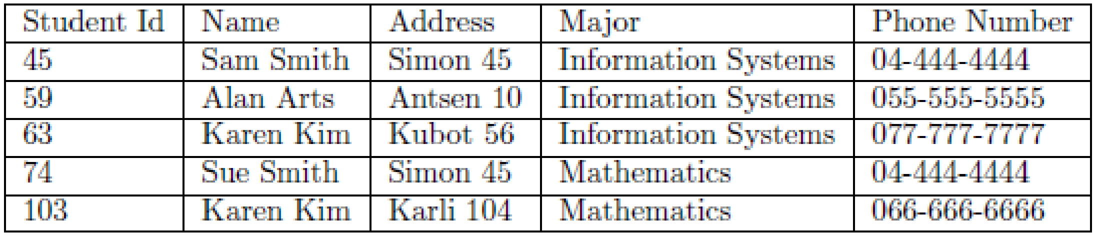
- Attributes must identify an entity in an entity set uniquely
  - The *Key* is the set of attributes which together uniquely identify each entity in the entity set
  - Other possibilities (also keys, but not in use) are called *Candidate Keys*
- Keys must be minimal
  - Why?

Attributes Example

- Back to the Students example:
  - Student Id
  - Name
  - Address
  - Degree Track (Major)
  - Phone Number



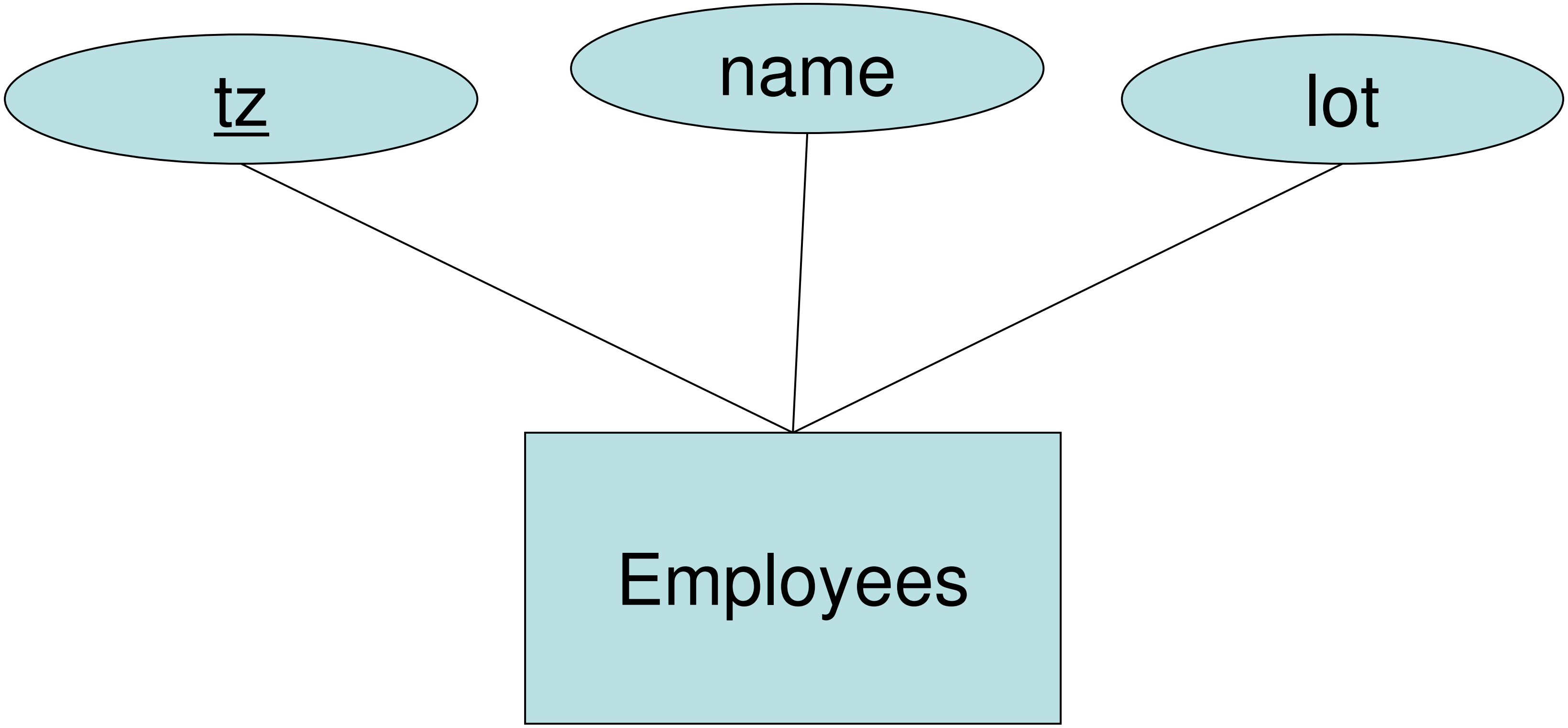
Student Id	Name	Address	Major	Phone Number
45	Sam Smith	Simon 45	Information Systems	04-444-4444
59	Alan Arts	Antsen 10	Information Systems	055-555-5555
63	Karen Kim	Kubot 56	Information Systems	077-777-7777



Student Id	Name	Address	Major	Phone Number
45	Sam Smith	Simon 45	Information Systems	04-444-4444
59	Alan Arts	Antsen 10	Information Systems	055-555-5555
63	Karen Kim	Kubot 56	Information Systems	077-777-7777
74	Sue Smith	Simon 45	Mathematics	04-444-4444
103	Karen Kim	Karli 104	Mathematics	066-666-6666

Employee Example and Diagram

- Employees have:
  - National Id
  - Name
  - Parking Lot



Relationships

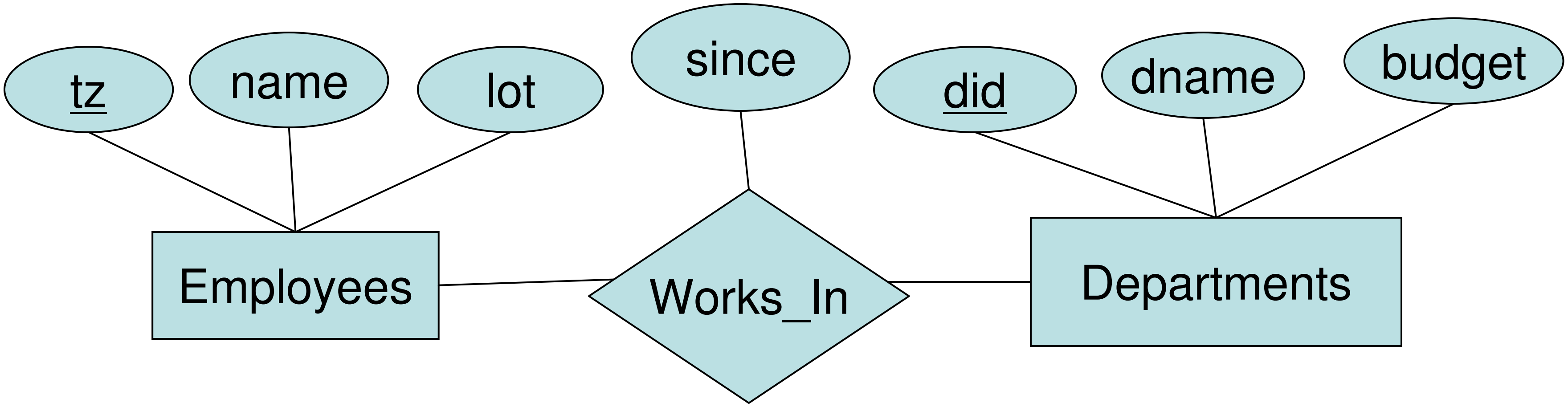
- A **relationship** is an association between two or more entities
  - Relationships can be sets too

- Formally, a relationship set is of n-tuples where each entity belongs to some entity set:

$$\{(e_1, \ldots, e_n) | e_1 \in E_1, \ldots, e_n \in E_n\}$$

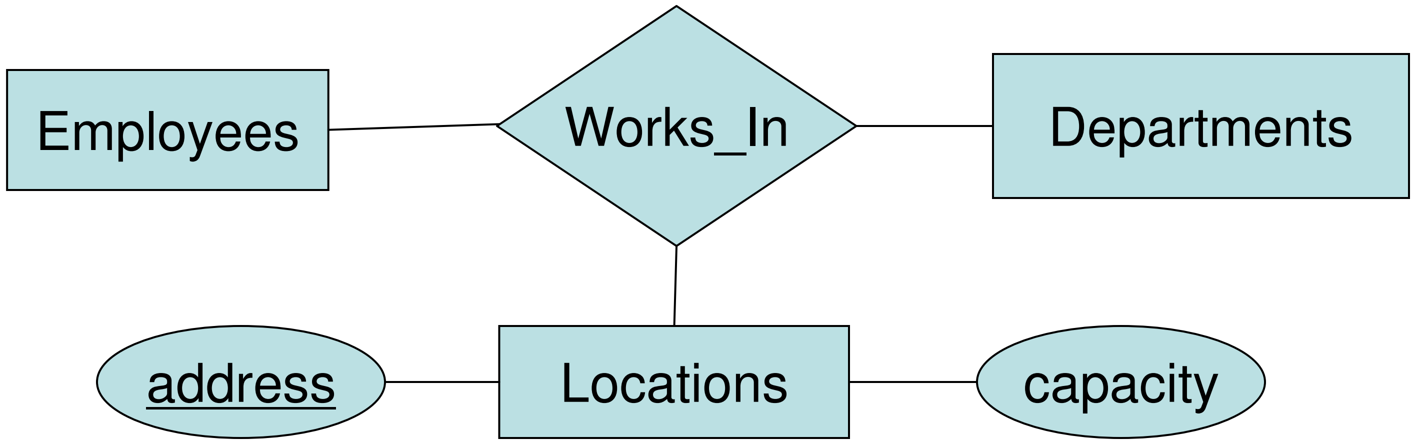
- This is read "A relationship is a set of tuples of the form *e_1*, *e_2*, …, *e_n* such that *e_1* is a member of entity set *E_1*, *e_2* is a member of entity set *E_2*, etc."

Relationship Example



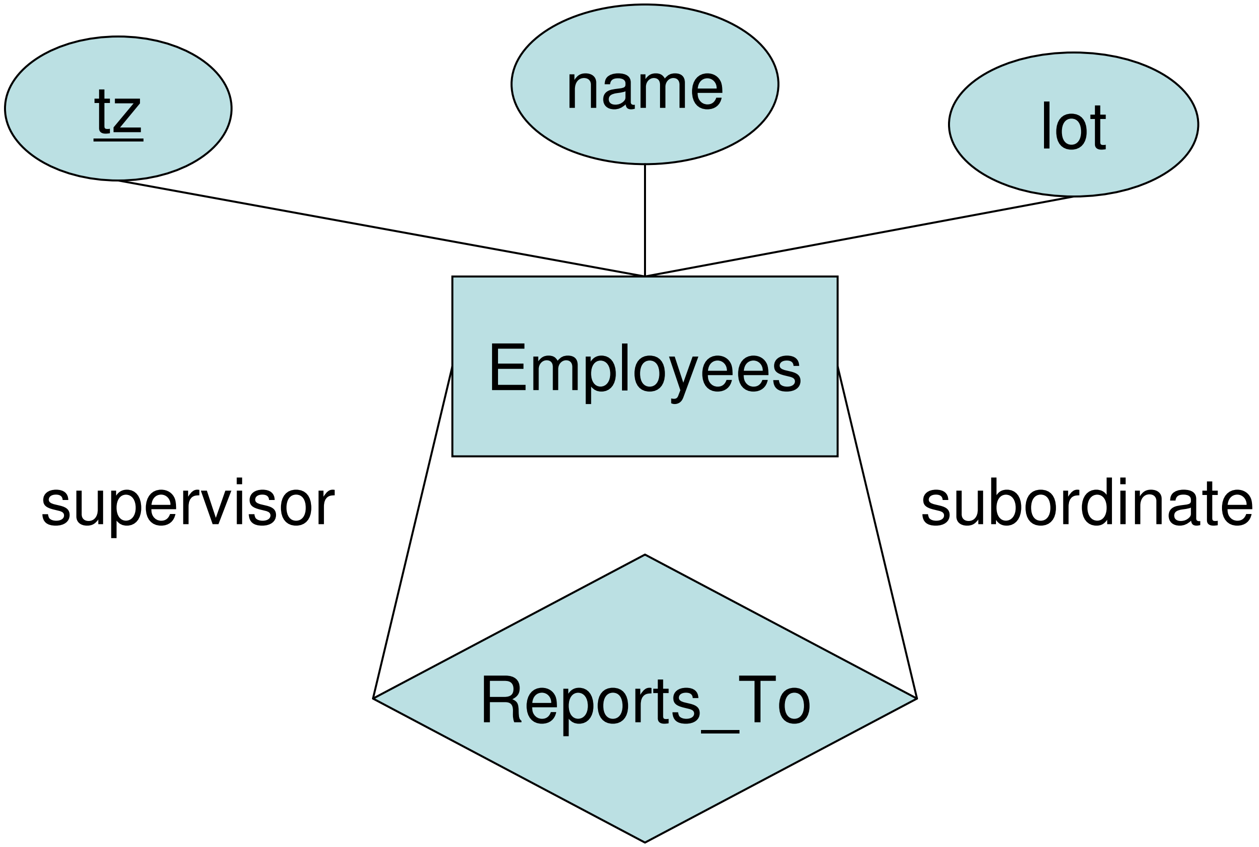
Relationships and Sets

- An **instance** of a relationship set is a relationship
- Relationships can binary:



- Or Ternary
- Or more

Reflexive Relationships



- Sometimes a role label can help identify important aspects of a relationship

So far

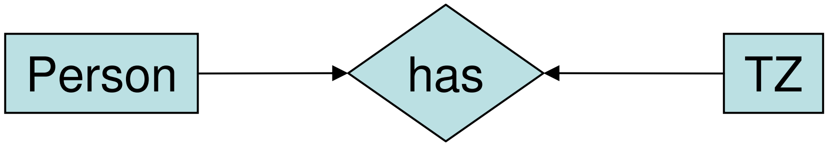
- Intro: Queries, Transactions, Crashes
- Structure of a DBMS
- Entity Relationship Diagrams
  - Simple to Complex
  - Entities, Attributes, Entity Sets
  - Relationships and Relationship Sets
  - Additional Features

Additional Features of ERD

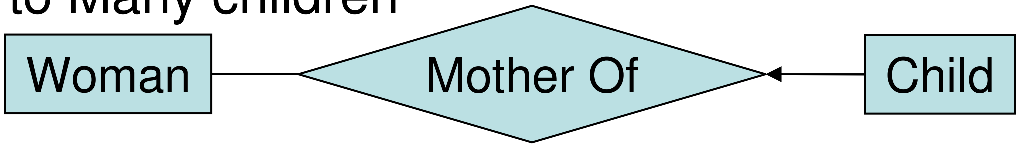
- ERDs can also represent more complex aspects of relationships:
  - Key Constraints
  - Participation Constraints
  - Weak Entities
  - Inheritance
  - Aggregation

Key Constraints

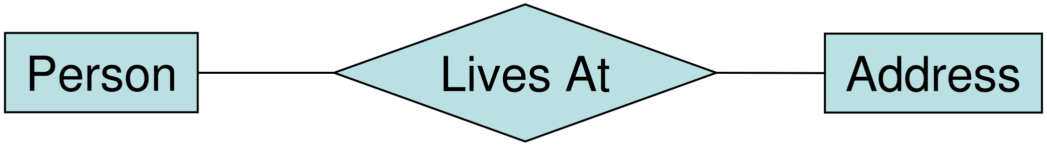
- Types of Key Constraints:
  - One to one
  - One to Many
  - Many to Many
- Examples:
  - One to One – 1 TZ to 1 person



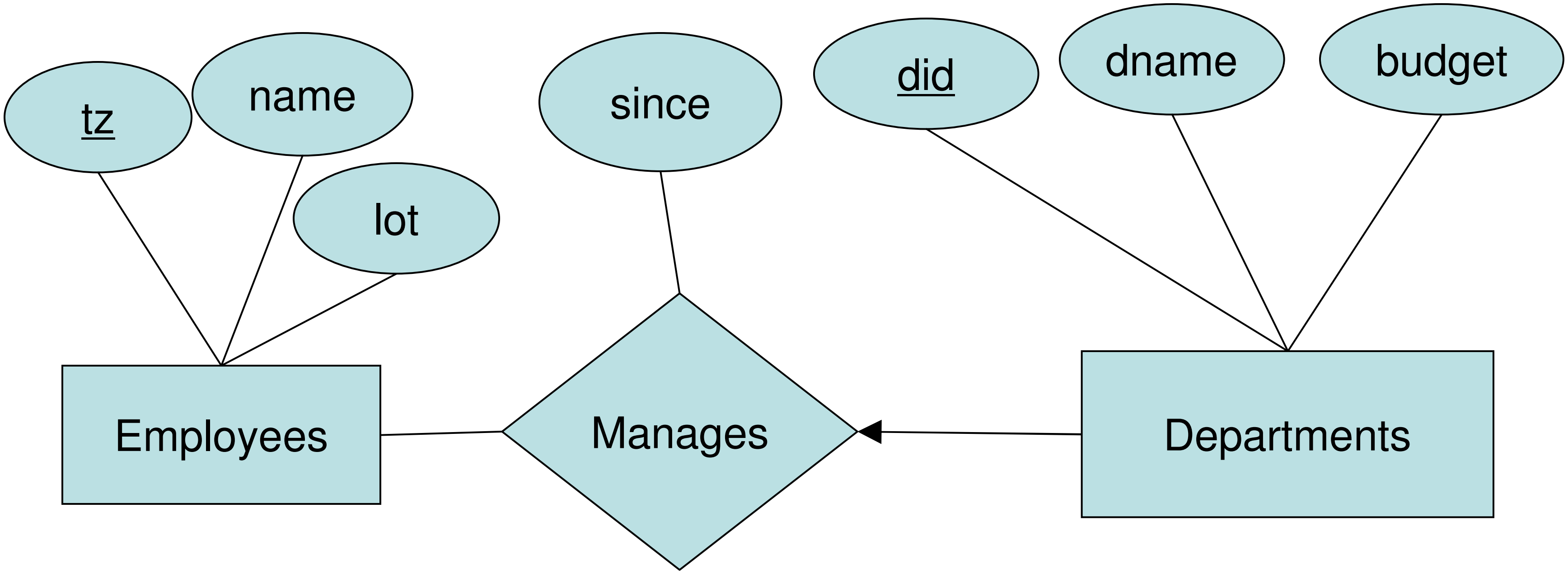
  - One to Many – 1 mother to Many children



  - Many to Many – Many addresses to Many residents

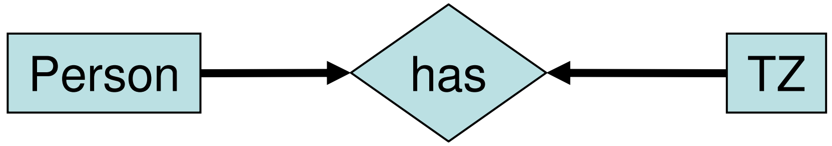


Key Constraint Example

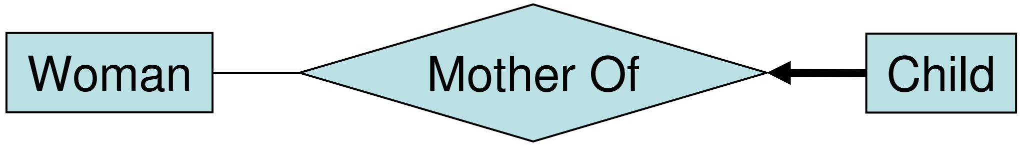


Participation Constraints

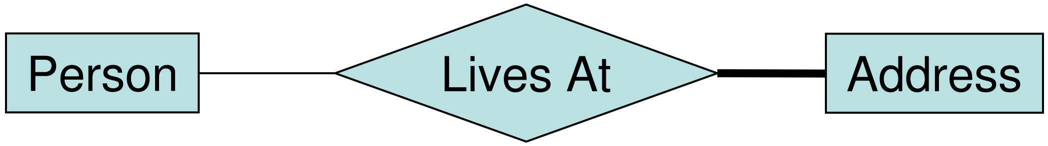
- Whether each member of an entity set must appear in the relationship
  - Thick line on its side if yes
- Examples:
  - Each person must have a TZ, every TZ is for a person



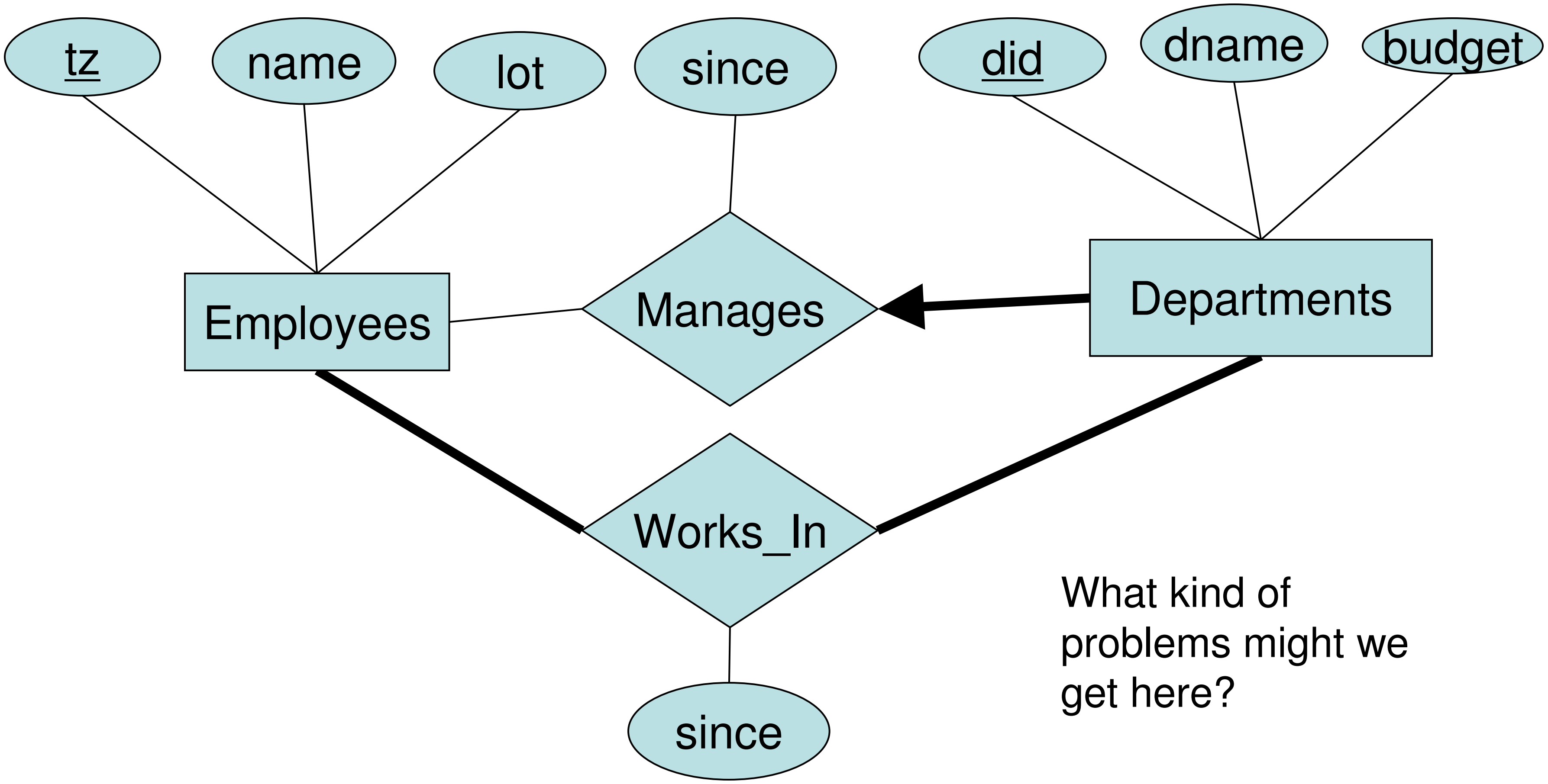
  - Each child must have a mother



  - Not everyone has an address, every address has someone there



Participation Example



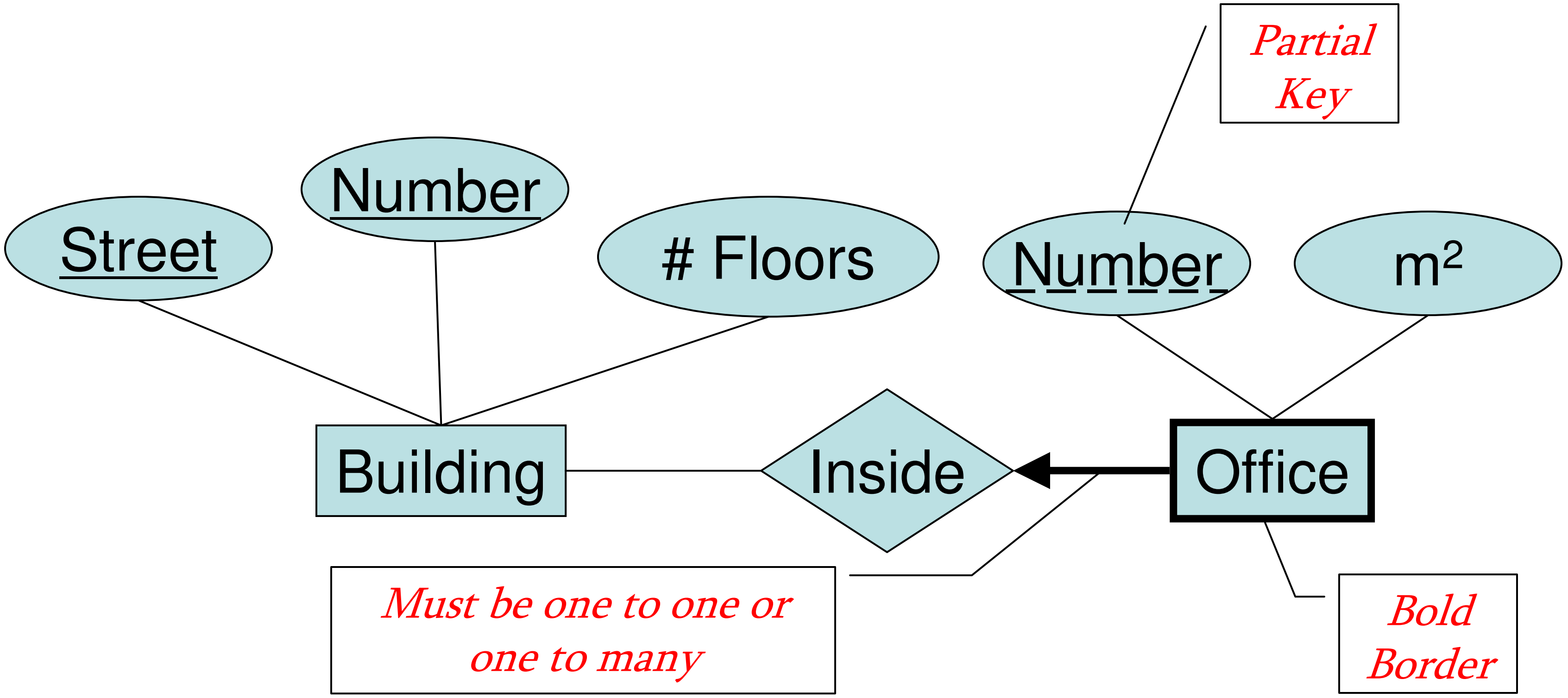
What kind of problems might we get here?

Weak Entities

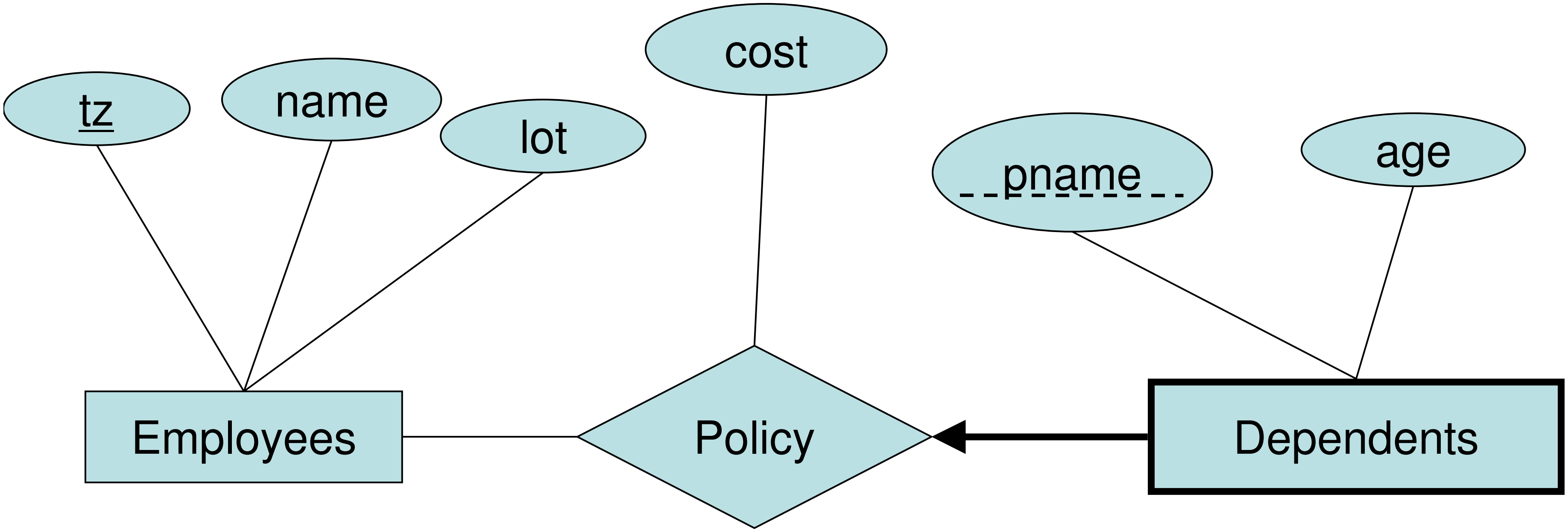
- These come up when there is something important which has no way of being identified except in terms of its relationship to another entity

- Example: We need to store data about offices in an office building
  - Each office has an office number and floor
    - ex. 308
  - Two offices in different buildings may have the same number
    - 308 in Building 1 and 308 in Building 2
  - We may identify an office based on its building

Weak Entities Example



Weak Entity Example



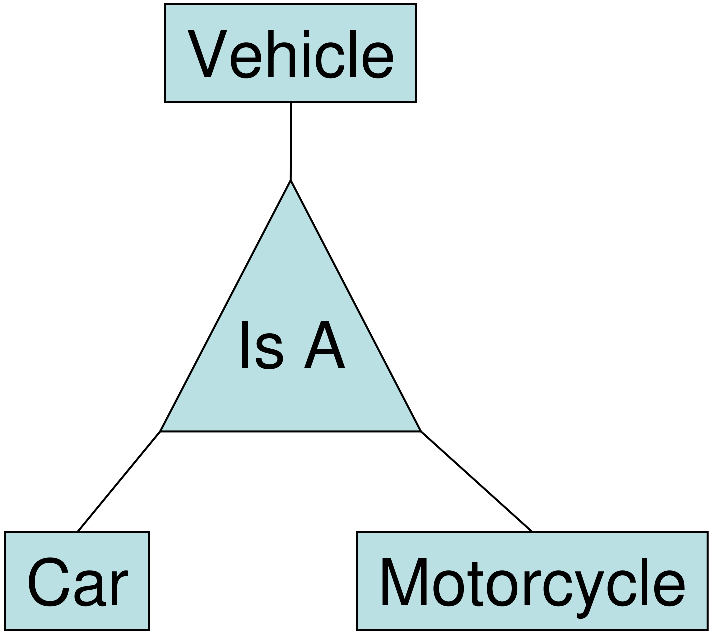
Inheritance

- ERDs can show hierarchical entity relationships
  - Inheritance in programming jargon
  - Limited to *Is A* relationships

- Examples:
  - A car *is a* vehicle
  - A motorcycle *is a* vehicle
  - A worker *is a* person
  - A manager *is a* person

- Has implicit containment semantics:
  - Every car/motorcycle entity is a vehicle as well
  - Every worker/manager is a person as well

Inheritance Examples

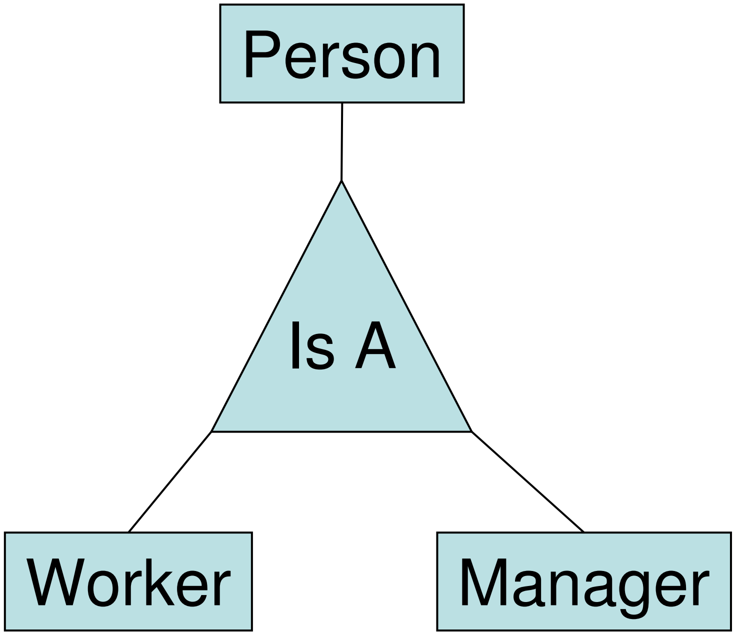


Vehicle

Is A

Car

Motorcycle



Person

Is A

Worker

Manager

Car and Motorcycle COVER Vehicle

Worker OVERLAPS Manager

Worker and Manager COVER Person

Conclusion

- Intro: Queries, Transactions, Crashes
- Structure of a DBMS

- Entity Relationship Diagrams
  - Simple to Complex
  - Entities, Attributes, Entity Sets
  - Relationships and Relationship Sets
  - Additional Features