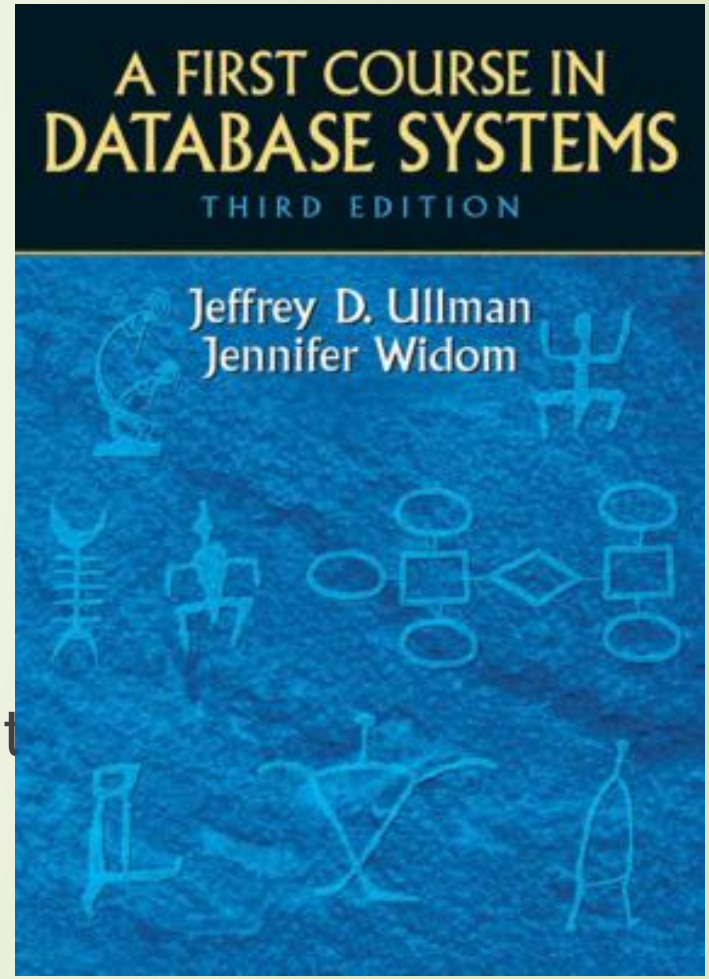
MPCS 53001: Databases
Fall 2017

Zach Freeman

Chicago Tribune

Class details

- TAs
  - Nirajan Khadga
  - Kathryn Smith
- Graders
  - JK Krug
  - Dushyant Srikant
  - Susie Yi
- Textbook (recommended)
  - A First Course in Database Systems (Ullman, Widom)



Class info/announcements

- Office hours (four times a week!*)
  - Check Piazza for specifics

* Once additional TA(s) is/are hired

Class sites

- Gradiance
  - Problem set homework system
  - http://www.newgradiance.com
  - Class token: **5F87D698**
- Piazza
  - Class management system
  - Use for ALL communications about the class
  - Lecture notes
  - https://piazza.com/uchicago/fall2017/mpcs53001
- Canvas
  - Project homework info/submission
  - https://canvas.uchicago.edu/courses/5134

Course work

- Weekly online homework (Gradiance)
  - Solve and submit online quizzes
- Weekly multipart project: database system (Canvas)
  - Design and develop a relational database and a web interface for it. Final product: a working web application with a MySQL backend.
  - Due before start of next class **(No late homework)**
- Midterm
  - In class 10/31 & 11/2.
  - Open book and notes (no laptops/devices/digital notes)

Lectures

- Tuesday 5:30-8:30 (Ryerson 276)
- Thursday 5:30-8:30 (Ryerson 251)
- No final exam
  - No class 11/21 & 11/23 (Thanksgiving week)
  - Makeup class Saturday 12/2 (9am-noon)
  - Final project due 12/5 & 12/7 (no class that week)

- Ask questions any time!

Class Overview

- Database design and modeling
  - E/R model, relational model, normal forms, relational algebra
- Database programming
  - SQL (language), MySQL (system)
- Web database programming
  - PHP
- Advanced topics
  - Data warehousing
  - NoSQL, big data, NewSQL

Guest speakers

- 11/28 & 11/30 – Datawarehousing
  - **Frank Greco**: Executive Software Client Architect/Executive Cloud Architect for IBM.
- 12/2 – Big Data/NoSQL
  - **Vincent Forgione**: lead engineer of Plenario within The University of Chicago's Urban Center for Computation and Data (UrbanCCD). Array of Things project.

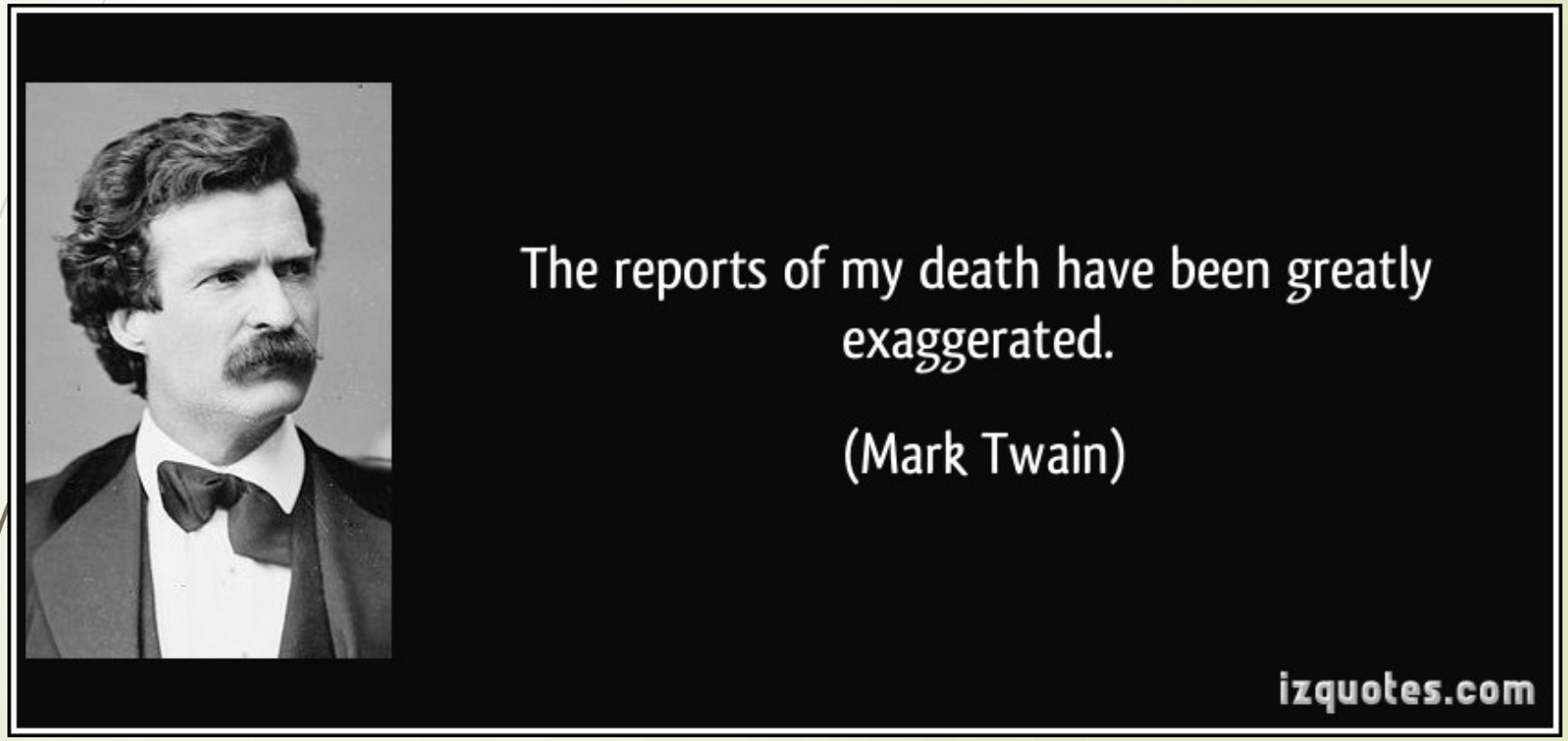
Final Project Examples

- From previous quarters

- Stype Planner
- Dungeons and Dragons

- NYC Community Resources
- NYC Community Resources Admin

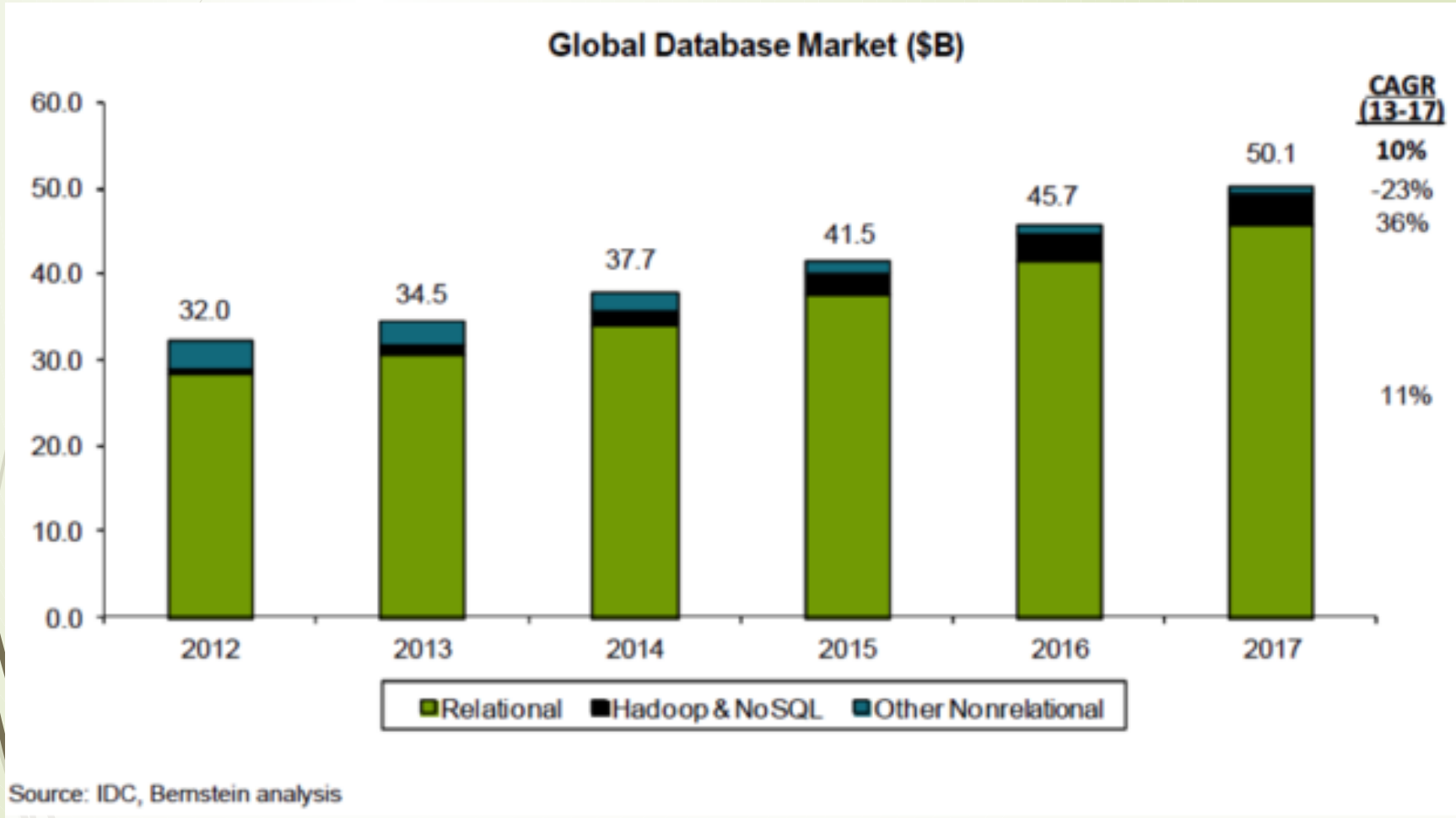
Isn’t everyone using NoSQL now?



MySQL 8.0 RC 1

- Released 9/25/2017: http://mysqlserverteam.com/mysql-8-0-rc1-highlights/

- NoSQL-centric highlights:
  - “It is important to have flexibility, and a large part of this story is schemaless JSON support.”
  - “The MySQL document store allows you to treat MySQL like a document database, with a set of NoSQL CRUD APIs to access your data.”



$50 billion market – 85-90% relational

Software Development Life Cycle (SDLC)

- Ideation/Requirements Gathering
- Design/Modeling
- Database creation/Fine-tuning
- Front-end creation/Fine-tuning
- Final product/Delivery

The Basics

Initial terminology

- **Data** - facts that are recorded and can be accessed
  - Data formats – text, numbers, figures, graphics, images, audio/video recordings, etc.
- **Information** - refers to the data that is accessed by a user for some particular purpose

- **Metadata** - data that describes the structure and the properties of the data
  - Metadata is essential for the proper understanding and use of the data

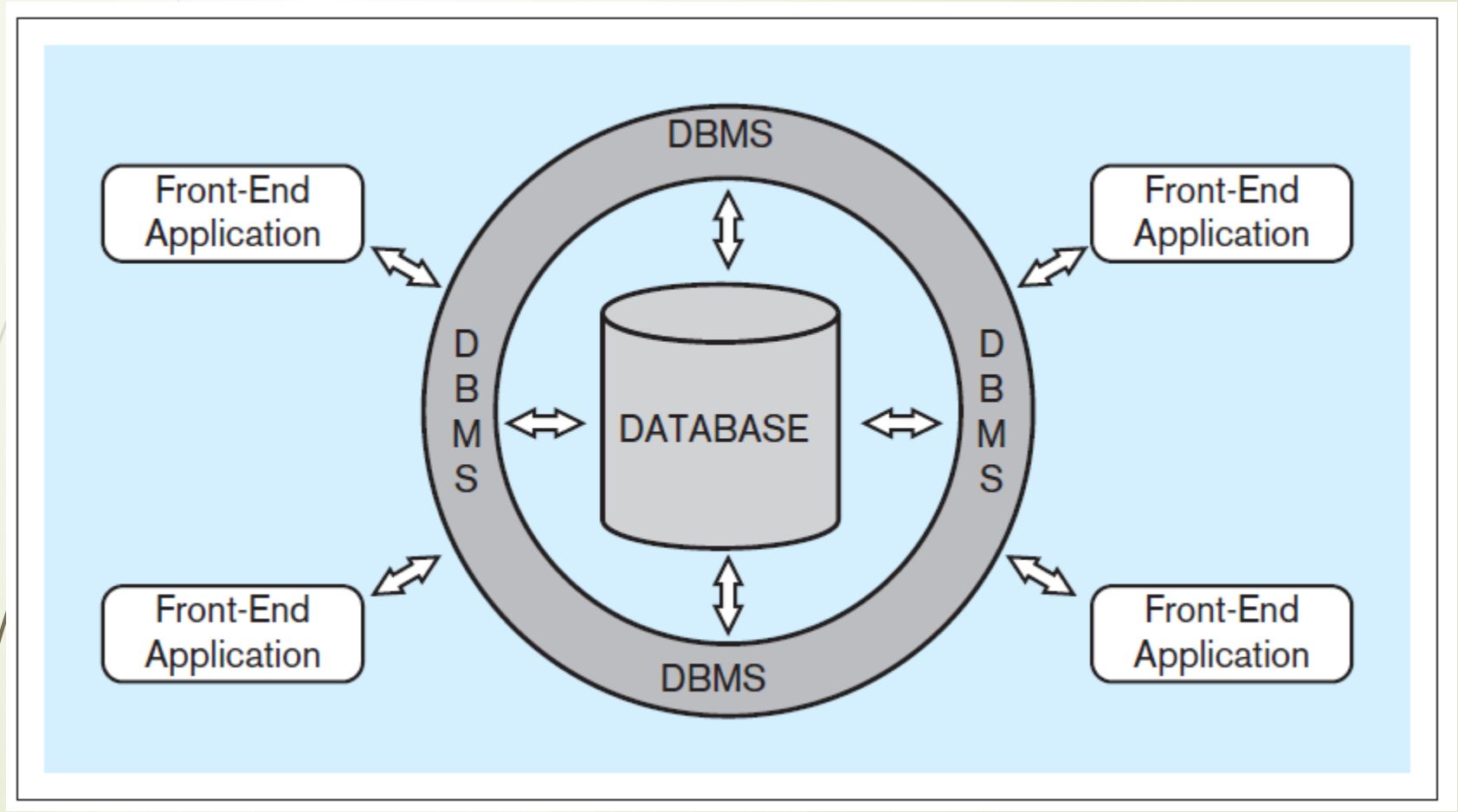
Initial terminology

- **Database** - structured collection of related data
  - Organizes the data in a way that facilitates efficient access to the information captured in the data

Database Management System (DBMS)

- A Database Management System – manages large amounts of data and provides:
  - Persistent storage
  - Efficient access
  - Concurrent access
  - Secure, atomic access
- Software used for:
  - Creation of databases
  - Insertion, storage, retrieval, update, and deletion of the data in the database
  - Maintenance of databases

Initial terminology



Databases are used by…

- Retailers
  - Amazon, Wal-Mart, etc…
- Reservation systems: travel, events, restaurants
  - Uber, Expedia, TicketMaster, etc…
- Libraries
- Social Media
  - Twitter, Facebook, Instagram, etc…
- Apps, websites, standalone structures (ATMs, RedBox, vending machines…)
- Voting machines (!!!)

Database System Roles

- **Database analysts, designers, and developers**
  - **Database analysts** - requirements collection, definition, and visualization stage
  - **Database designers** (a.k.a. **database modelers** or **architects**) - database modeling stage
  - **Database developers** –implement the database model as a functioning database using the DBMS software

Database System Roles

- **Front-end applications analysts and developers**
  - **Front-end application analysts** – collect and define requirements for front-end applications
  - **Front-end application developers** - create the front-end applications (CSS/javaScript/HTML)

Database System Roles

- **Database administrators (DBAs) -** perform the tasks related to the maintenance and administration of a database management system

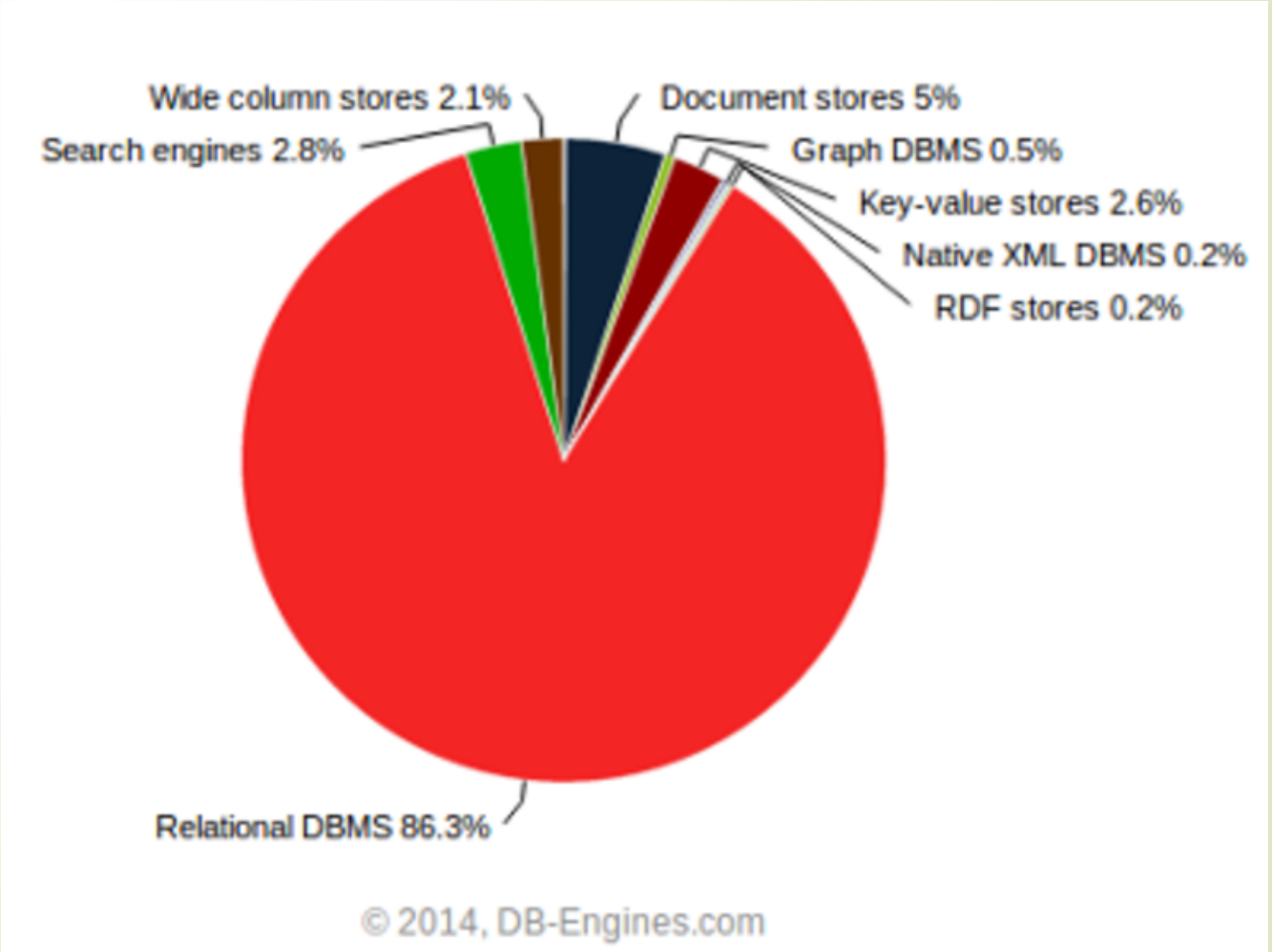
Database System Roles

Relational DBMS History

- Earlier models: hierarchical, network
- Relational proposed by Ted Codd at IBM in 1970
- Relational algebra enables high-level query language development
  - Separates querying (data retrieval) from implementation and storage

Relational DBMS History

- Since 1990 relational DBMS have dominated the database marketplace



Database marketplace

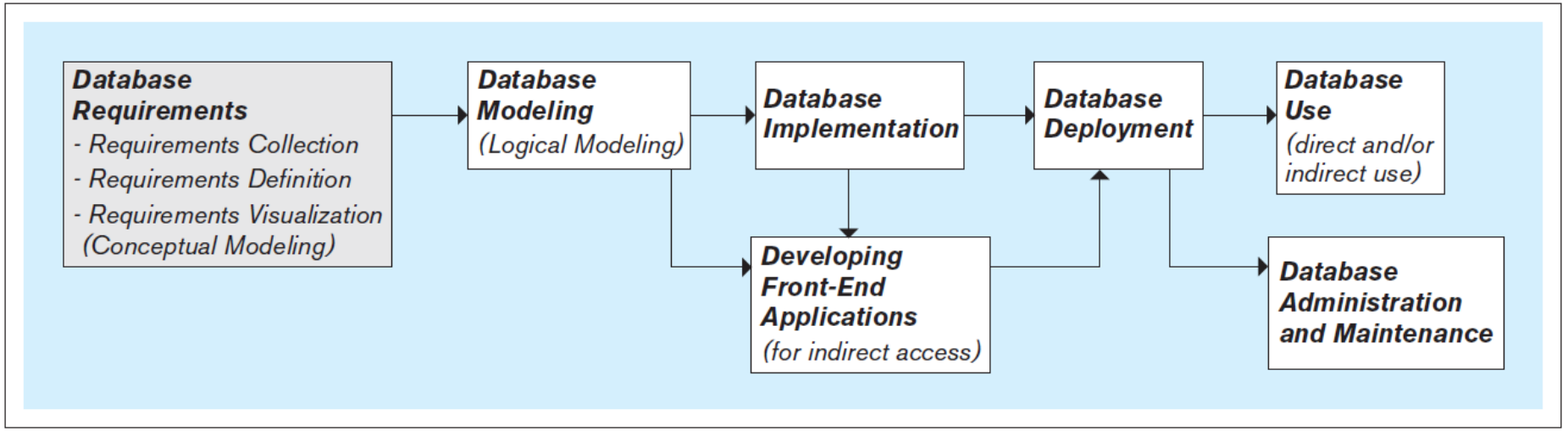
- Big three vendors:
  - Oracle
  - Microsoft (SQL Server)
  - IBM (DB2)
- Many others:
  - MySQL (owned by Oracle), PostgreSQL (open source)
  - Teradata, Greenplum (warehousing)
  - Cassandra, MongoDB, CouchDB (NoSQL)
  - Spanner, VoltDB (NewSQL)

Studying DBMS

- Modeling and design
  - Explore issues before implementation, ideate, gather requirements, model
- Programming
  - Create structure (tables, views, stored procs)
  - DB operations: queries, updates, connectivity
- DBMS implementation
  - Storage, access, querying speed, indexing

- Our focus is on 1 and 2.

Steps in the development of database systems



Steps in the development of database systems

- **Requirements collection, definition, and visualization** - results in the requirements specifying which data the future database system will hold and in what fashion, and what the capabilities and functionalities of the database system will be
  - The **collected** requirements should be clearly **defined** and stated in a written document, and then **visualized**

The benefits of planning

- **"If I had eight hours to chop down a tree, I'd spend six sharpening my axe."**

Entity-relationship model

- **Entity-relationship (ER) modeling** - conceptual database modeling technique
  - Enables the structuring and organizing of the requirements collection process
  - Provides a way to graphically represent the requirements

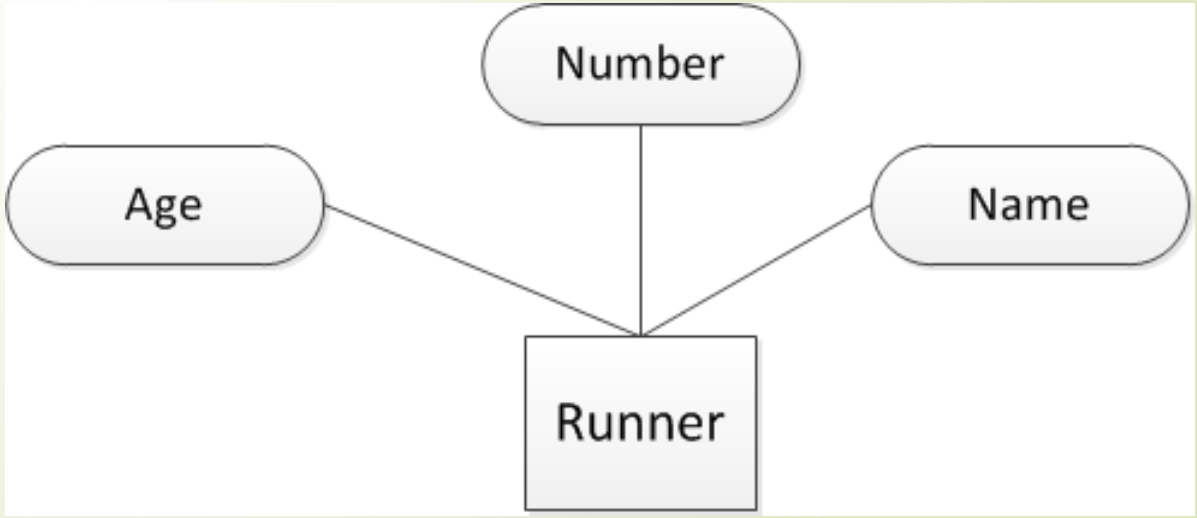
- **ER diagram (ERD)** - the result of ER modeling
  - Serves as a blueprint for the

Entity-relationship model

- First step of DB design
  - Represent real-world scenario with diagrams
- Key concepts:
  - Entity = object
  - Entity set = collection of similar objects
  - Attribute = property of entities in entity set
  - Relationship = connection between entity sets

e/r diagrams

- Entity set: rectangle
- Attribute: oval



Entity sets as tables

- The entities in an entity set are typically stored as rows in a table:

- Runner table:

Name	Age	Number
Zach Freeman	34	125
Nirajan Khadga	32	138
Kathryn Smith	26	245

‹ OCTOBER 7, 2017

1

Last Week: 2

Bodak Yellow (Money Moves)
Cardi B

New

2

Last Week: --

Rockstar
Post Malone Featuring 21 Savage

3

Last Week: 1

Look What You Made Me Do
Taylor Swift

4

Last Week: 3

1-800-273-8255
Logic Featuring Alessia Cara & Khalid

5

Last Week: 4

Despacito
Luis Fonsi & Daddy Yankee Featuring Justin Bieber

Relational databases

- Example of data organized in a relational table

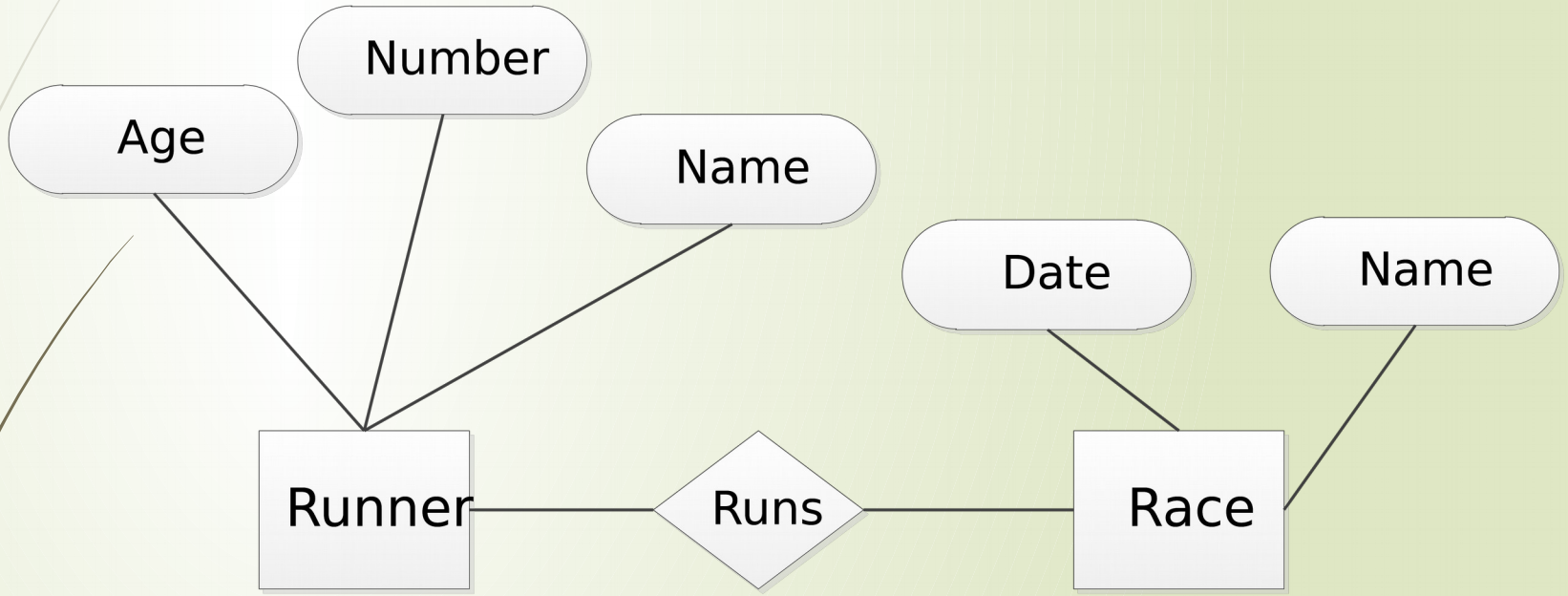
Billboard100 table possible schema:

Artist	**Title**	**Rank**	**LastWeek**
Cardi B	Bodak Yellow (Money Moves)	1	2
Post Malone	Rockstar	2	--
Taylor Swift	Look What You Made Me Do	3	1
Logic	1-800-273-8255	4	3
Luis Fonsi & Daddy Yankee	Despacito	5	4

Relationships

- **Relationship -** ER modeling construct depicting how entity sets are related
  - Within an ER diagram, each entity set must be related to at least one other entity set via a relationship

Relationships



Relationship sets

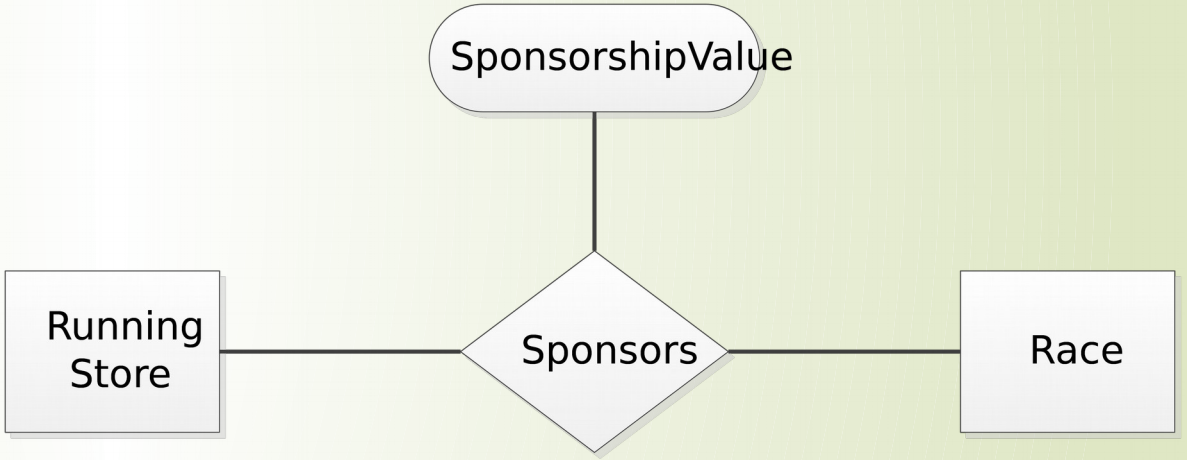
- The value of a relationship set is the set of connected entities:
  - Think of it as a table with one row for each connection and one column for each connected entity set.

Runner	Race
Zach Freeman	Climb Hot Springs
Nirajan Khadga	Frozen Gnome 10K
Kathryn Smith	Cherry Blossom 10 Miler
J K Krug	Chicago Marathon
Zach Freeman	Arkansas Traveller 100

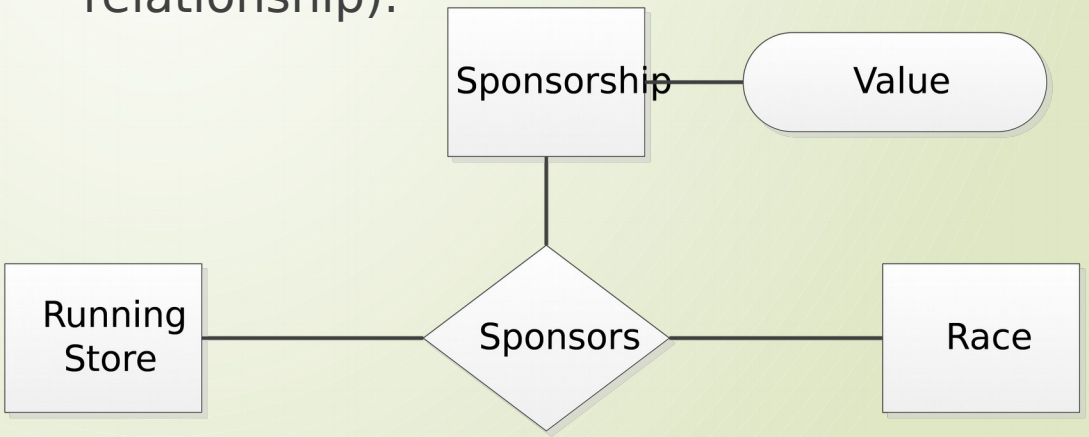
Multi-way relationships

- Binary relationships are most common
  - And easiest to implement
- Sometimes we need a relationship connecting 3 or more entity sets.
- Example: Runners, Races, Awards

Attributes on relationships



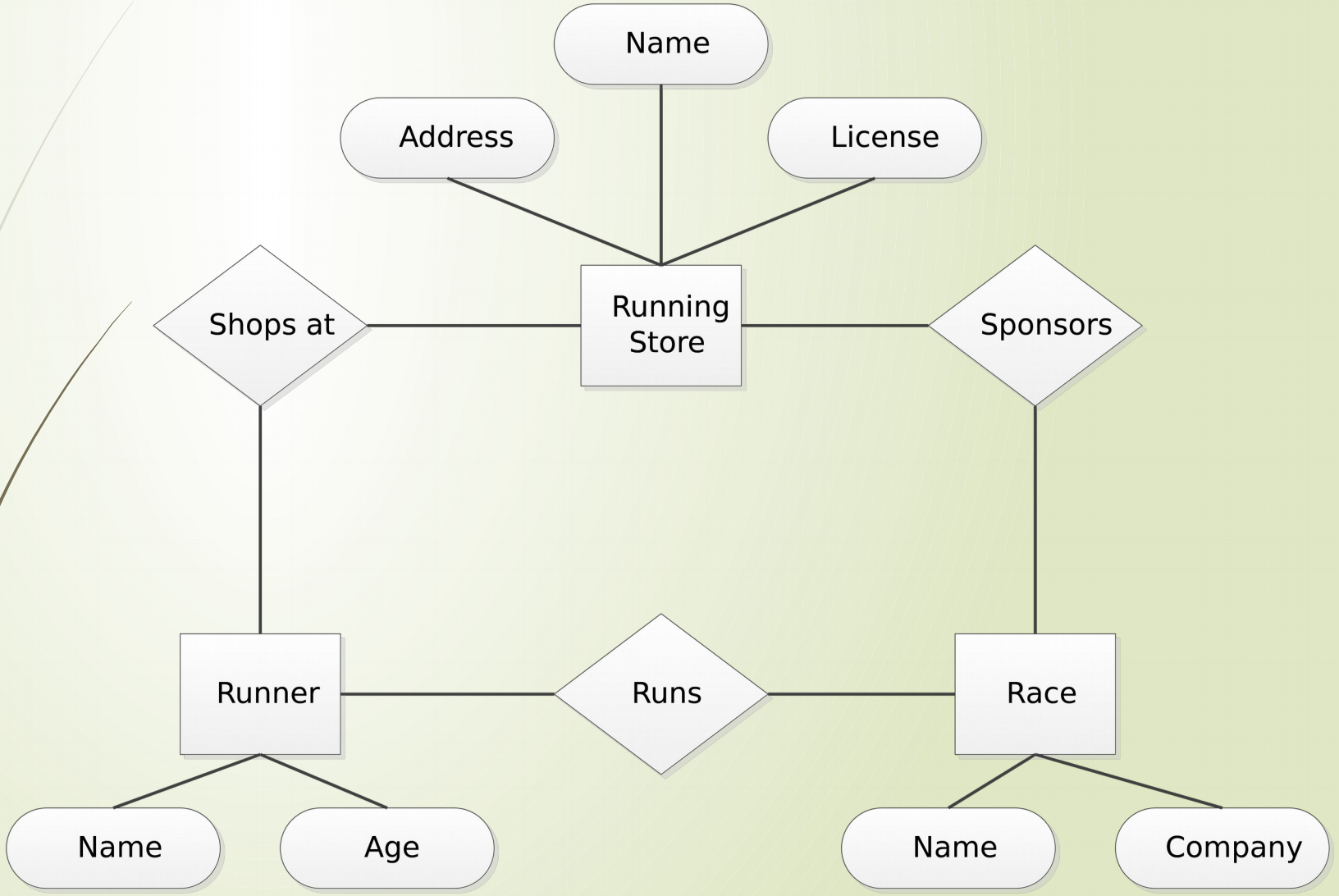
- Moving attribute to entity set (3-way binary relationship).



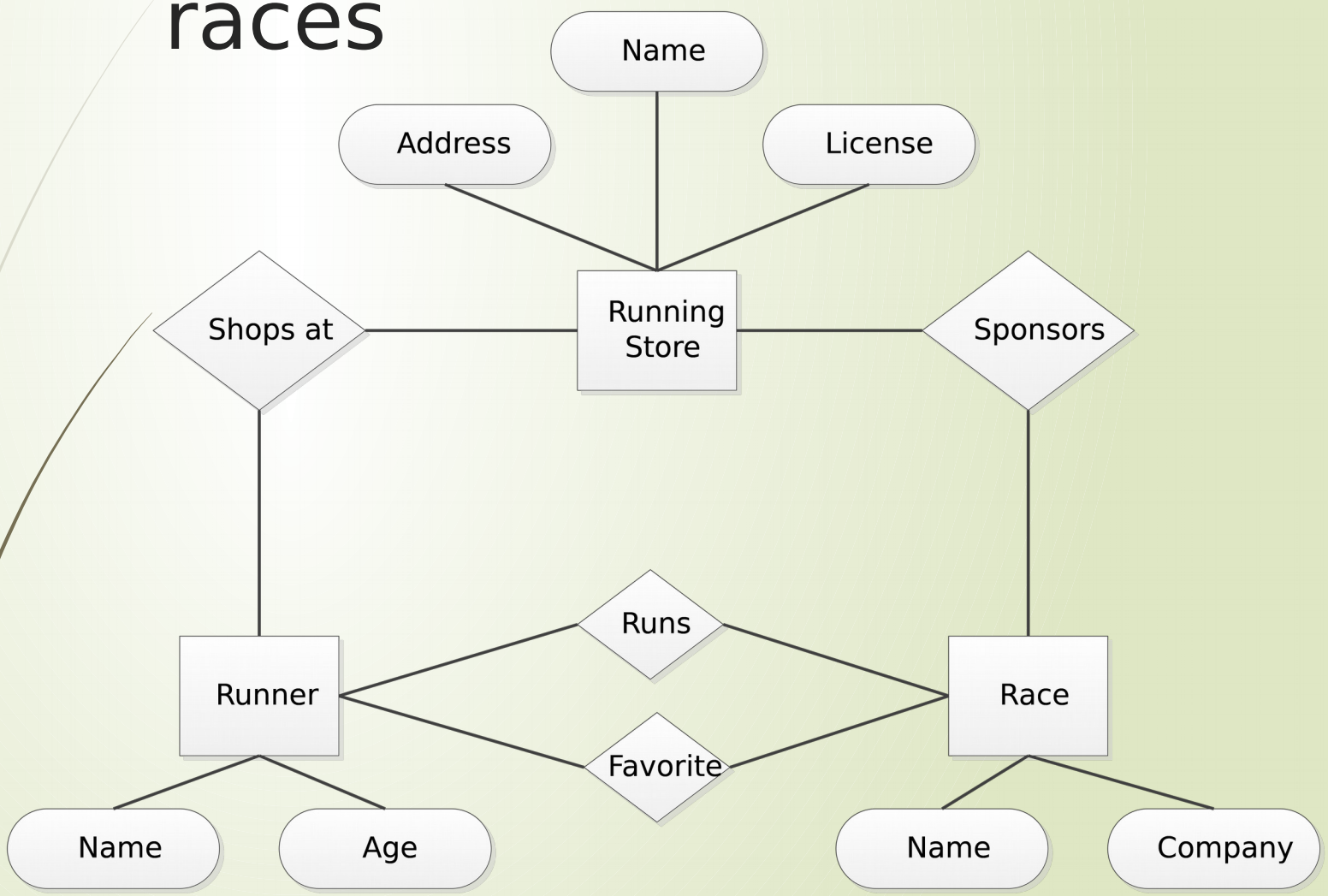
Multiplicity of relationships

- Mostly for binary relationships. Tricky for multi-way relationships.
- Many-to-many
  - No restrictions
- Many-to-one
  - Restricted on one side to a single connection per object.
- One-to-one
  - Restricted on both sides to a single connection per object.

Runner-Race-Store example

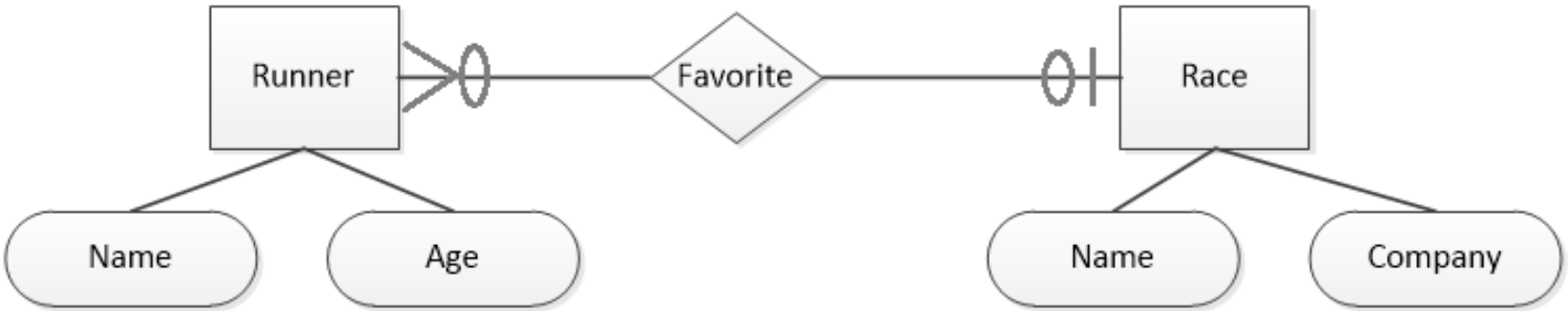


Runners have favorite races

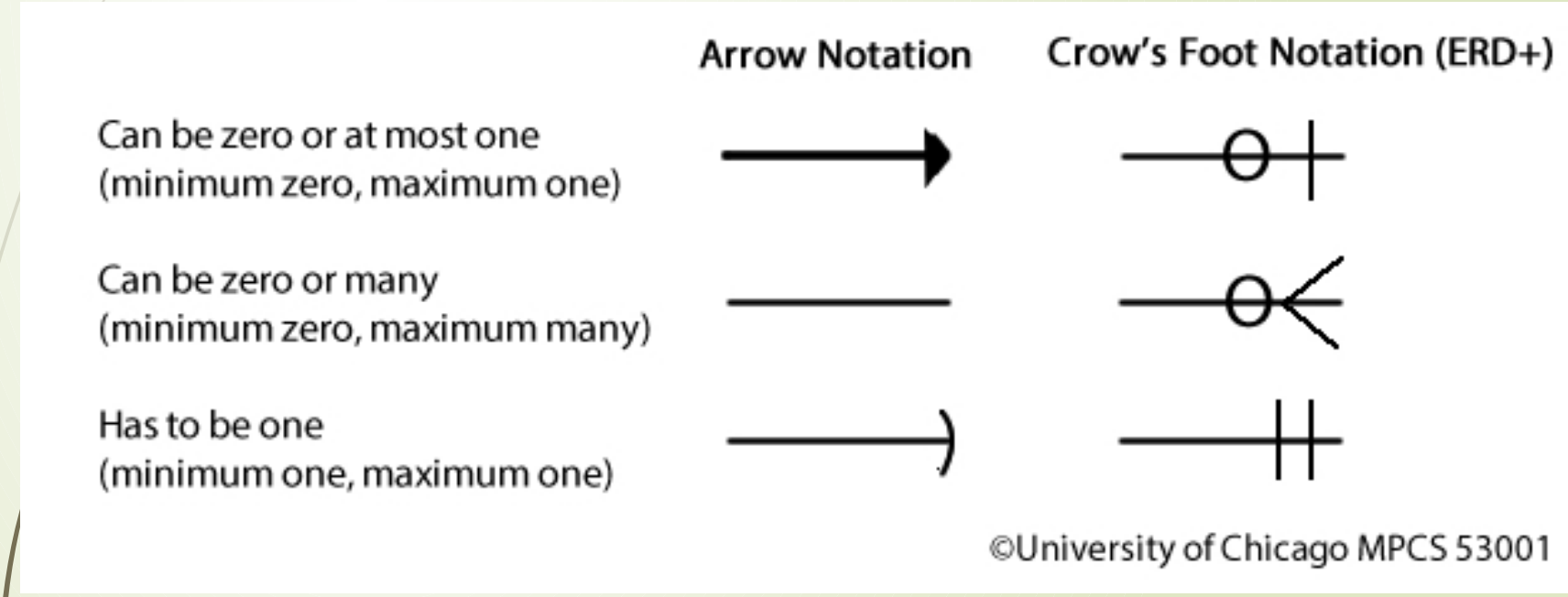


Crow’s foot notation

- Designating multiplicity in relationships



Crow’s foot vs. Arrow



One-One relationships

- At most one connection per entity.
- Arrows in both directions.

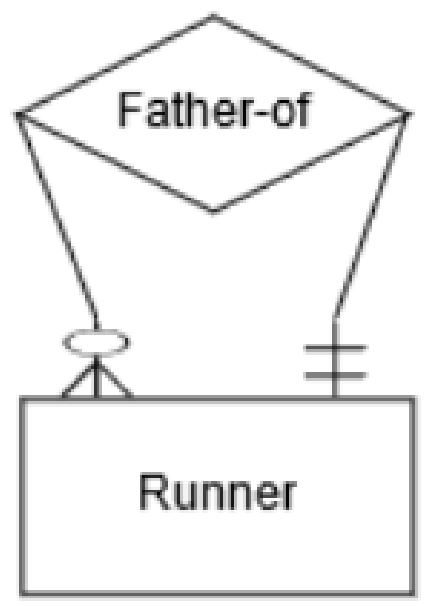
- Design issues:
  - Is the rounded arrow (exactly one) justified?
  - Should Company be an entity set or just an attribute?

One-One relationships

- Same diagram in crow’s foot:

Relationship roles

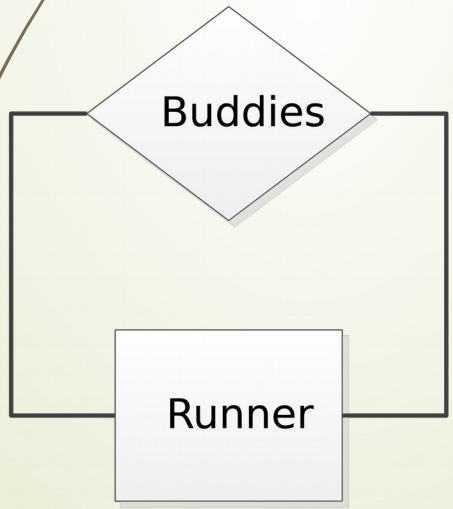
- An entity set can participate in a relationship more than once
- Labels can be added to distinguish roles



Father	Child
Brian	Van
Antonio	Avi
Jack	Lila

Running buddies

- Buddies is symmetric, Father-of is not.
- Cannot specify symmetric in E/R.



Buddy1	Buddy2
Nirajan	Bill
Kathryn	Susie
Anh	Serguei

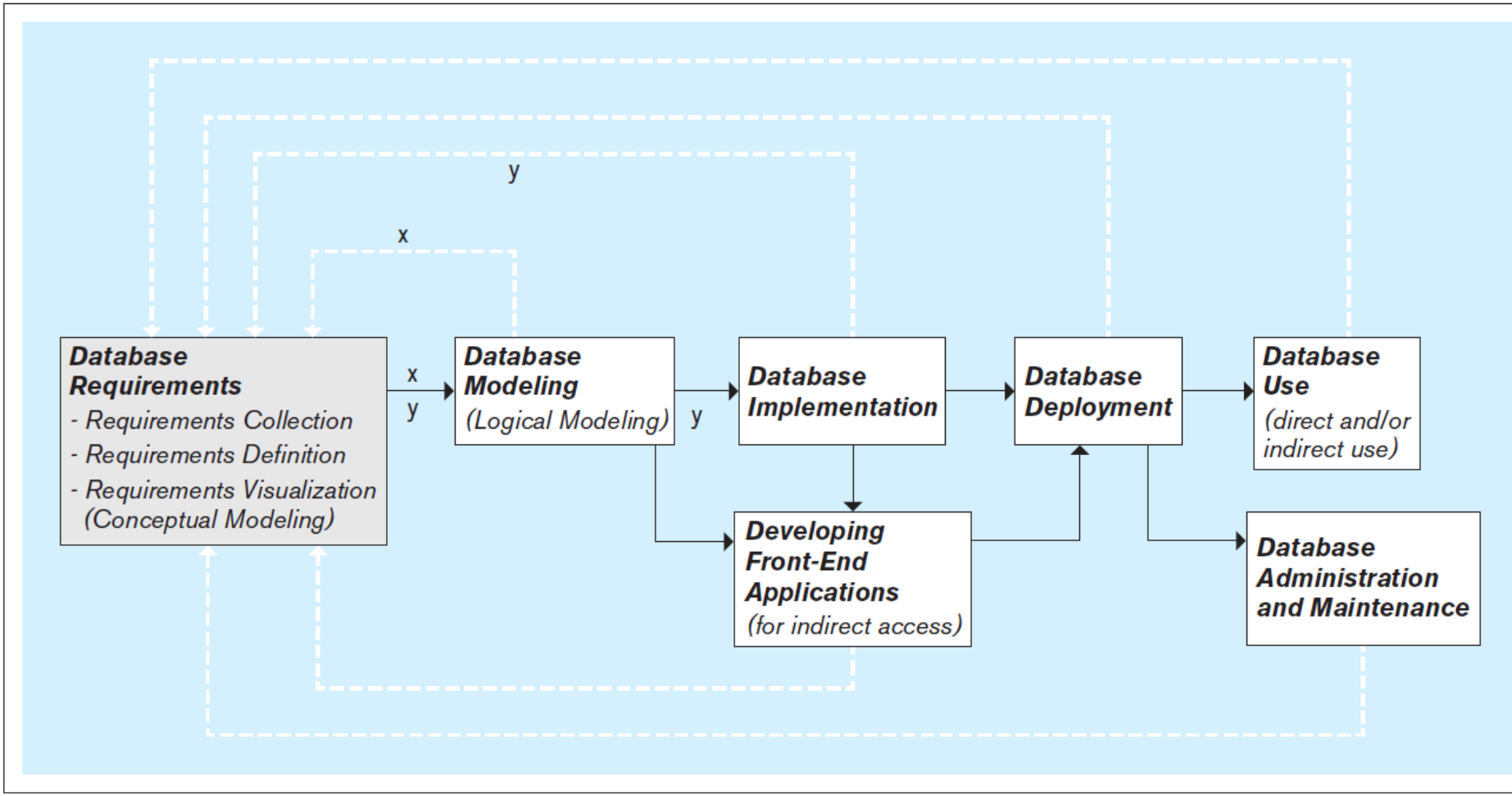
The big picture

- Stages of building a DB application:

1. Understand client needs (documentation)
2. Design data model (E/R diagram)
3. Convert data model to relational schema
4. Create structure (tables, etc.) in DBMS and load data

DB Development Steps

Database requirements collection, definition and visualization process is **iterative**



More design considerations

- Subclasses
- Keys
- Weak entity sets
- Key design principles
- Constraints

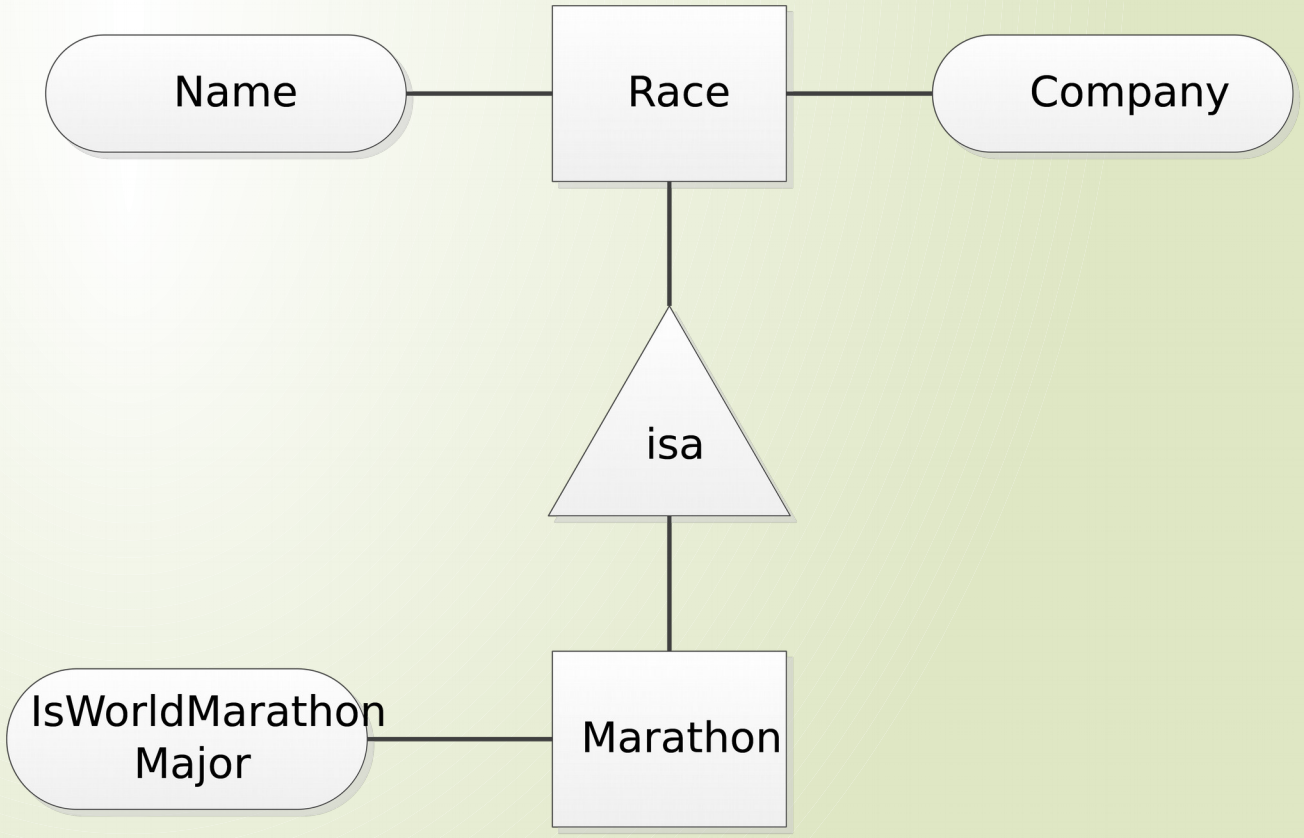
Subclasses

- Subclass characterization
  - Special case
  - Fewer entities
  - More properties (attributes + relationships)

- *Example*: Marathons are a kind of race. In addition to all race properties (attributes), they have **additional** properties.

E/R subclasses

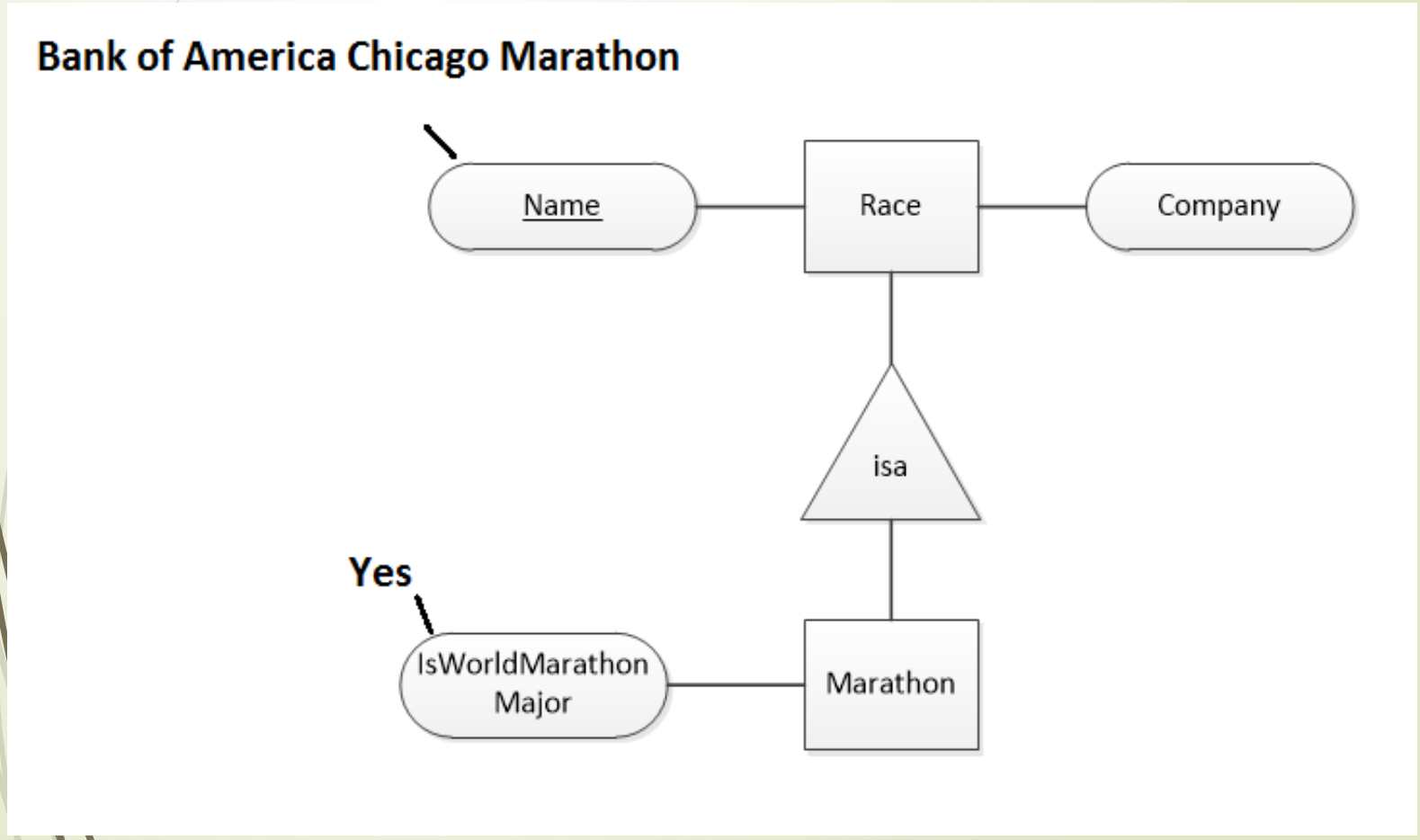
- Isa triangles indicate subclass relationships



E/R subclass interpretation

- An entity has a component in each entity set to which it belongs.
  - Its properties are the union of the properties of these entity sets.
  - e.g. Marathon has attributes in both Marathon and Race entity sets.
- Contrast with object-oriented view: an entity belongs to exactly one class and inherits properties of its superclasses.
  - e.g. Marathon would inherit attributes from Race.

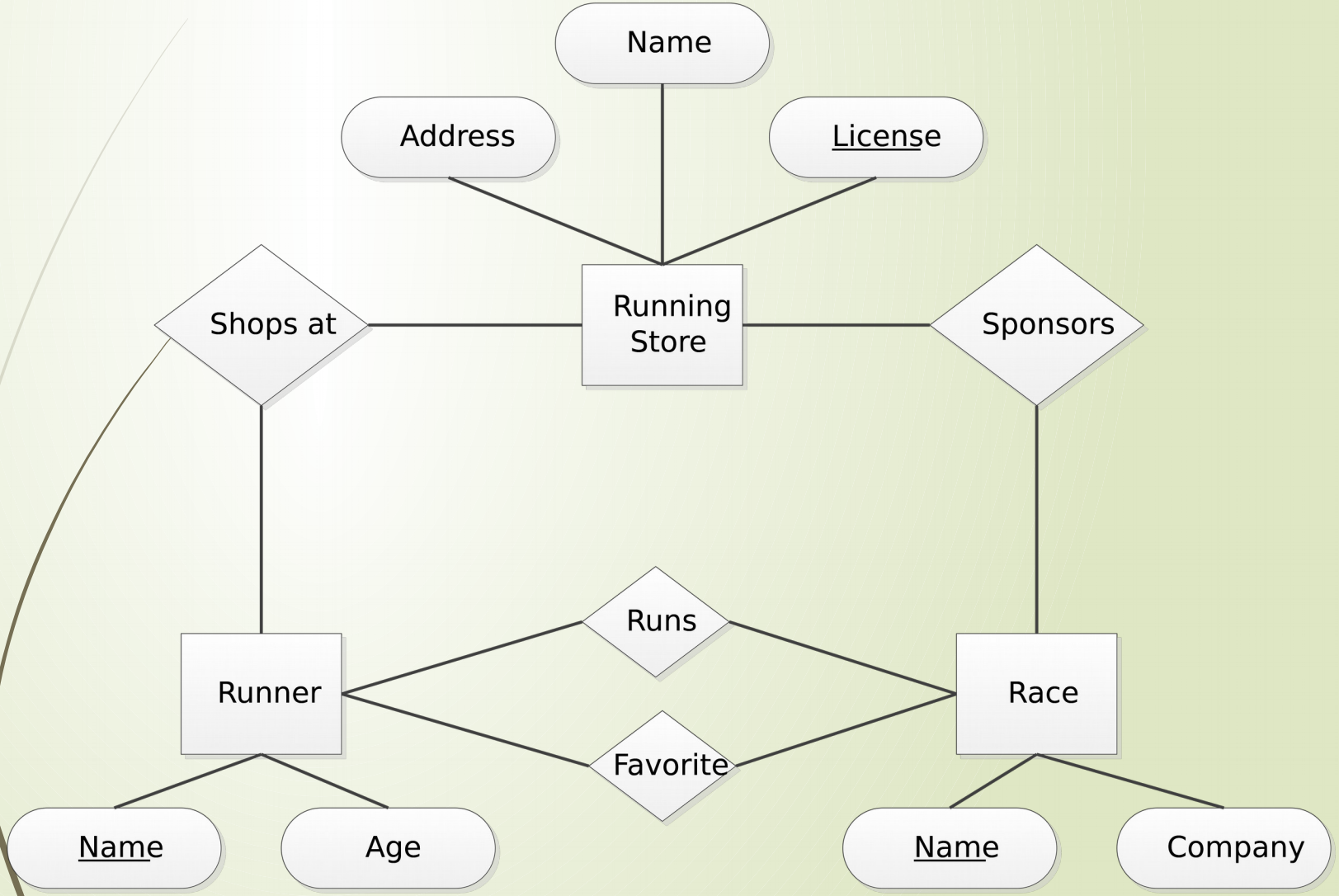
Subclass example



Keys

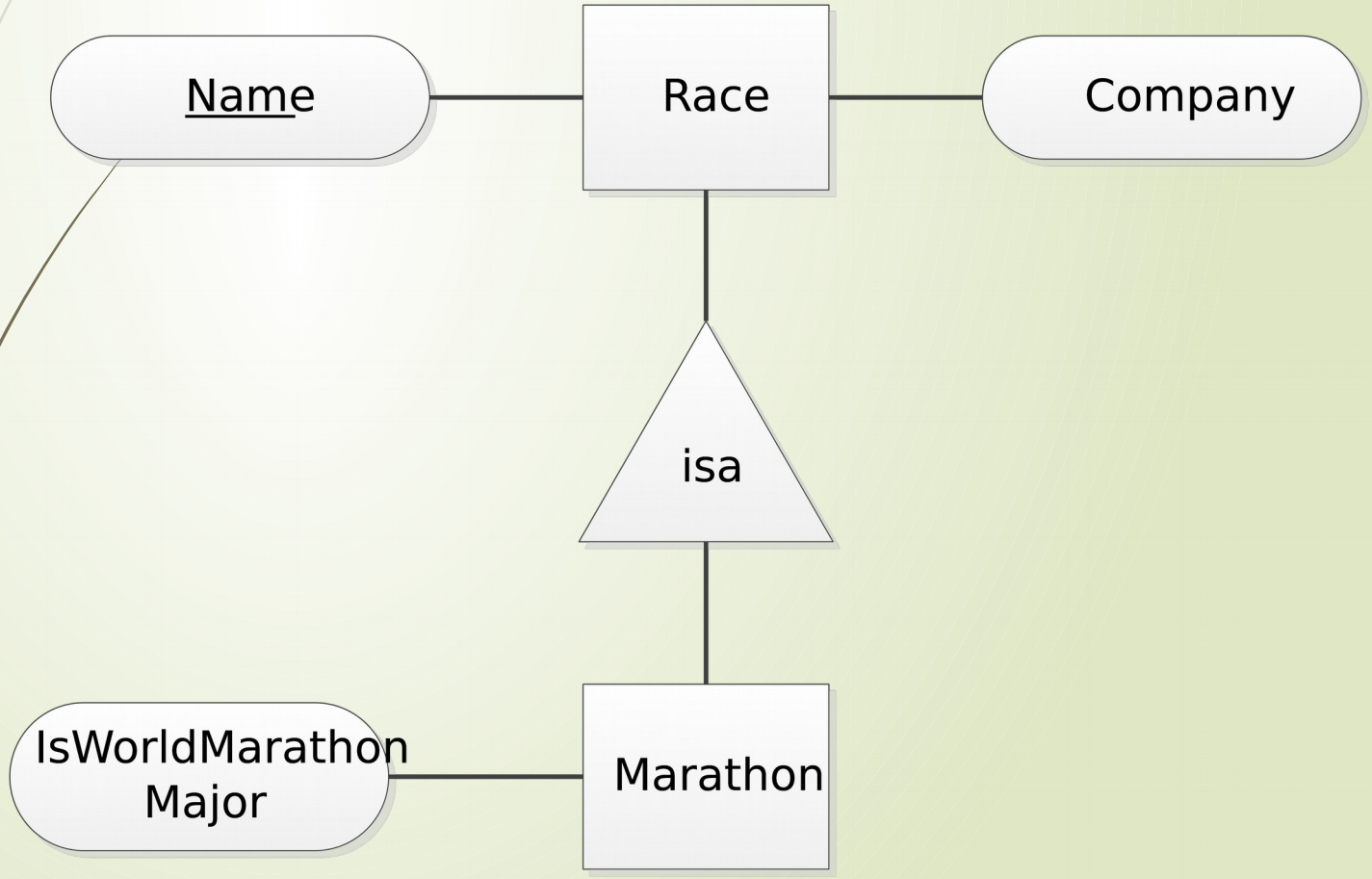
- A set of attributes whose values can belong to at most one entity
  - **Uniquely identifies an entity**
- In E/R model, every entity set must have a **primary** key
  - Can have more than one key, but one set of attributes is the **primary** key.
- Attributes of the primary key should be underlined.

Runner-race-store keys



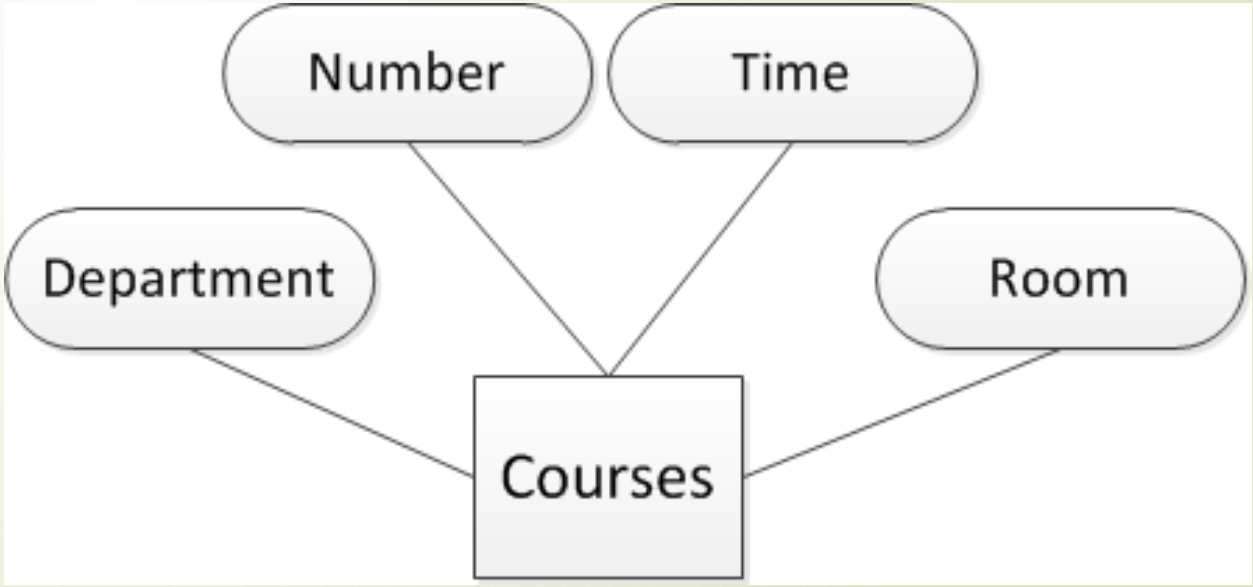
Subclass keys

- The key at the root is the key for all



Multi-attribute key

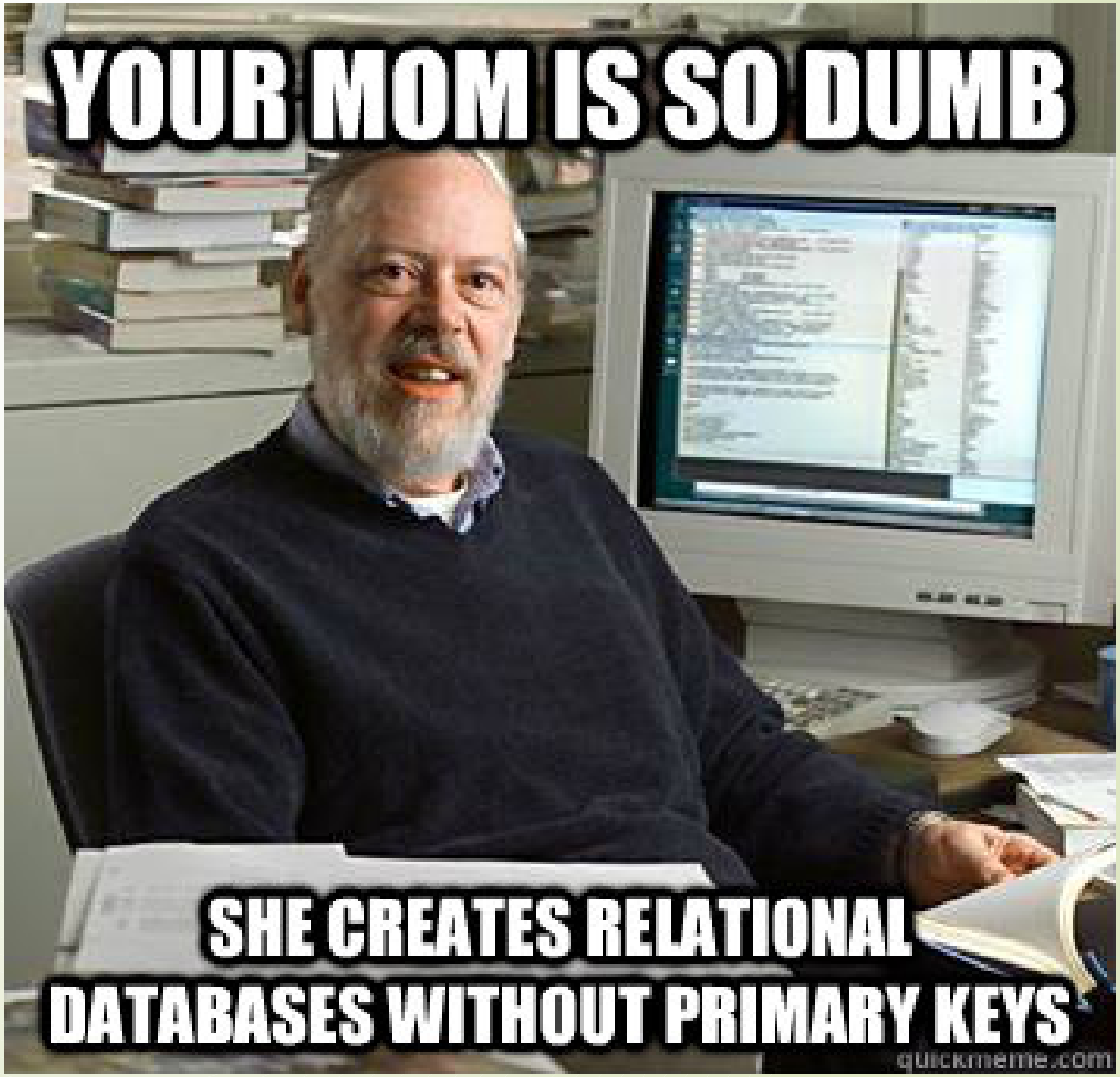
- What are all the possible keys?



Surrogate keys

- Synthetically generated unique identifiers, often by the DBMS.
- Pros: immutable, faster performance
- Cons: disassociation, ambiguity
- Implementation:
  - AUTO_INCREMENT (MySQL, MS SQL)
  - SERIAL (PostgreSQL)
  - SEQUENCE (Oracle)

Keys = Super Important



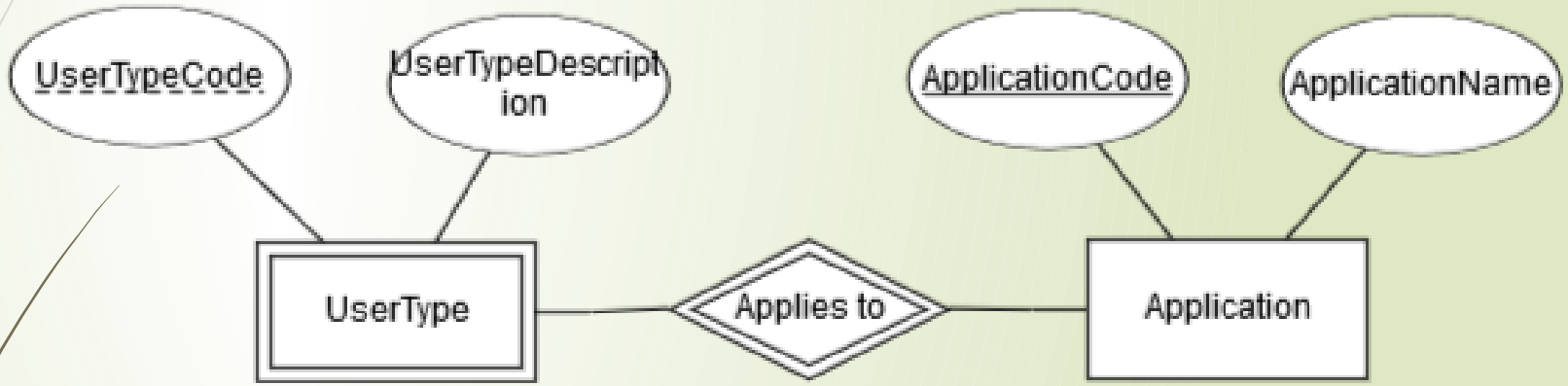
Weak entity sets

- For some entity sets, keys come from their own attributes AND from the keys of one or more other entity sets.
  - The first entity set must be connected to the other entity sets by a many-one relationship.
- The entity set is called **weak**
  - Relationships to other entity sets contributing to its key are called **supporting**.
- **Double rectangles** represent weak entity sets and **double diamonds** = supporting relationships.

Weak entity set example

- User Type = usertypecode + applicationcode
  - e.g. ADMIN | GRANTAPP
- Given an application, usertypes are unique
- Key for usertypes: usertypecode (which is unique for that application only) + the application code (unique globally)

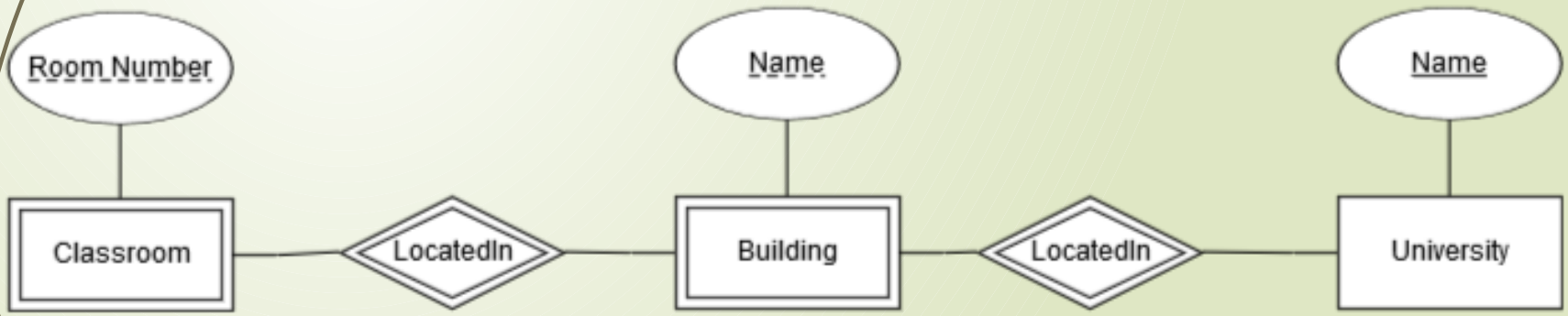
User Types



- Design issue: How could we make UserTypes stand alone and get rid of weak entity sets?

Chain of weakness

- Key for university = its name
- Key for building = its name + name of university
- Key for room = its number + name of building + name of university



Constraints

- Integral part of the database schema. Specify restrictions on the actual data.

- Keys
- Single value constraints
- Referential integrity constraints
- Domain constraints
- General constraints

Constraint details

- Single value constraint: each attribute has a single atomic value.
  - No set attributes (lists)! Use relationships instead.
- Referential integrity: the values of some attributes must come from among the values of another.
  - Implicit in E/R model from the relationship definitions.

Design principles

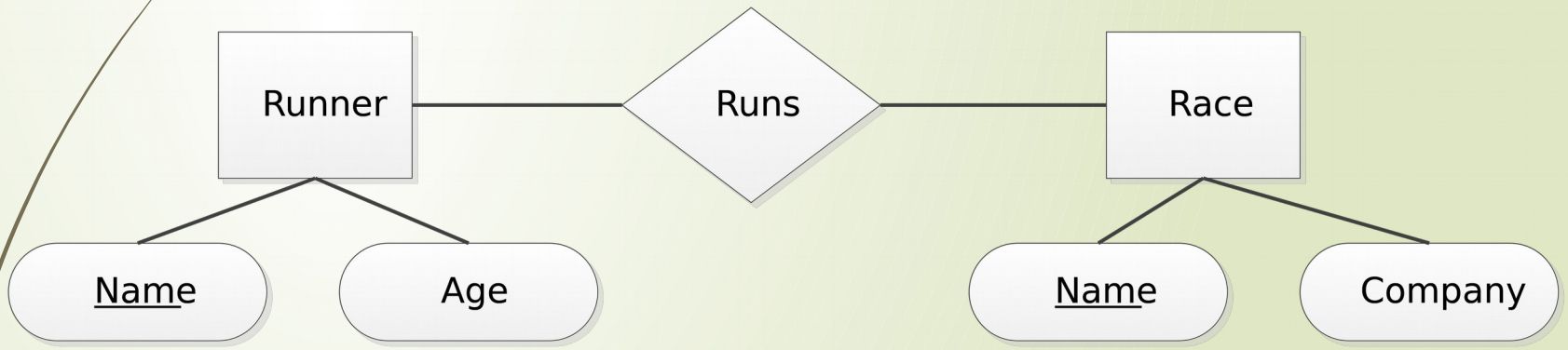
- Your clients have a (pretty good) idea of what they **want**. You have to design a database that represents exactly what they **need** (and what you need to be able to give them that).

- Avoid redundancy!
  - Represent each fact exactly once.
  - Reduces storage, avoids inconsistency.

Reflect Requirements

- Design schema should represent (and enforce) as many constraints as possible.
- Do not rely on data to follow assumptions.
- *Example:* “one instructor per course” should lead to a many to one relationship in your E/R diagram.

Design choices



- Can we represent the same information with a single entity set?

Summary

- Relational DBMS
- E/R model
  - Entities, entity sets, attributes, relationships
  - Multiplicity of relationships
  - Subclasses
  - Weak entity sets
  - Keys
- Design considerations
  - Principles
  - Requirements
  - Constraints

Next time

- Relational model
- Converting an E/R diagram to relations
- Database design theory
  - Functional dependencies (FD)
  - Normalization

First Homework

- Details in Canvas

- Practice with ERDPlus
- Enroll in class in Piazza
- First Gradiance problem set
- Come up with project idea.


QUESTIONS?
COMMENTS?
CONCERNS?