MPCS 53001: Databases

Zach Freeman

Lecture #2

Class announcements

- Office hours – on Piazza (Resources✉Staff).
  - Monday, Tuesday, Wednesday, Thursday (hours differ slightly)

- Assignment 2:
  - 2^{nd} Gradiance problem set
  - 2^{nd} part of final project (ERD, relational model)

Today's Class

- E/R modeling examples
- Relational model
- Converting E/R diagrams to relations
- Database design theory
  - Functional dependencies (FD)

Relational model

- Mapping ER Diagrams into Relational Schemas:
  - After ER diagram is constructed, it is then mapped into a relational schema

- Relation (entity set **OR** relationship) = table
- Attribute = column = field
- Tuple = row = record

Introduction

- **Relation -** table in a relational database
  - A table containing rows and columns
  - Main construct in the relational database model
  - Every relation is a table, not every table is a relation
  - Example: Singles table

Artist	Title	Rank	LastWeek Rank
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

Schemas

- Relational schema
- Example: Single(Title, Artist, Rank, LastWeekRank, VideoLink, CoverImage, etc…)
- Attribute order is **arbitrary**
  - In SQL order matters; default order is given by the relation definition
- Database schema
  - All relational schemas in a database

Relational tables

Order doesn’t matter (of columns or rows)

Rank	Title	LastWeekRank	Artist
5	Despacito	4	Luis Fonsi/Daddy Yankee
3	Look What You Made Me Do	1	Taylor Swift
2	Rockstar	--	Post Malone
4	1-800-273-8255	3	Taylor Swift
	Bodak Yellow (Money		

Relational vs. Non-

Relational Table (Relation)

EmpID	EmpName	EmpGender	EmpPhone	EmpBdate
0001	Joe	M	x234	1/11/1985
0002	Sue	F	x345	2/7/1983
0003	Amy	F	x456	4/4/1990
0004	Pat	F	x567	3/8/1971
0005	Mike	M	x678	5/5/1965

Not a Relational Table

EmpID	EmpInfo	EmpInfo	EmpPhone	EmpBdate
0001	Joe	M	x234	1/11/1985
0002	Sue	F	x345	2/7/1983
0001	Joe	M	x234	1/11/1985
0004	Pat	F	x567, x789	3/8/1971
0005	Mike	M	x678	a long time ago

Instances

- Relation instance: a set of tuples in a relation
  - Current instance: the set of tuples currently in a relation
- Updates change the current instance
- Most databases do not keep track of instance changes beyond the boundaries of a transaction
  - Changes can be stored explicitly by the application (in other tables in the database) log
- Database instance: all relation instances in a database

Why relations?

- Simplicity
- Comprehensibility
- Theoretical foundations
  - Relational algebra
- Efficient implementation

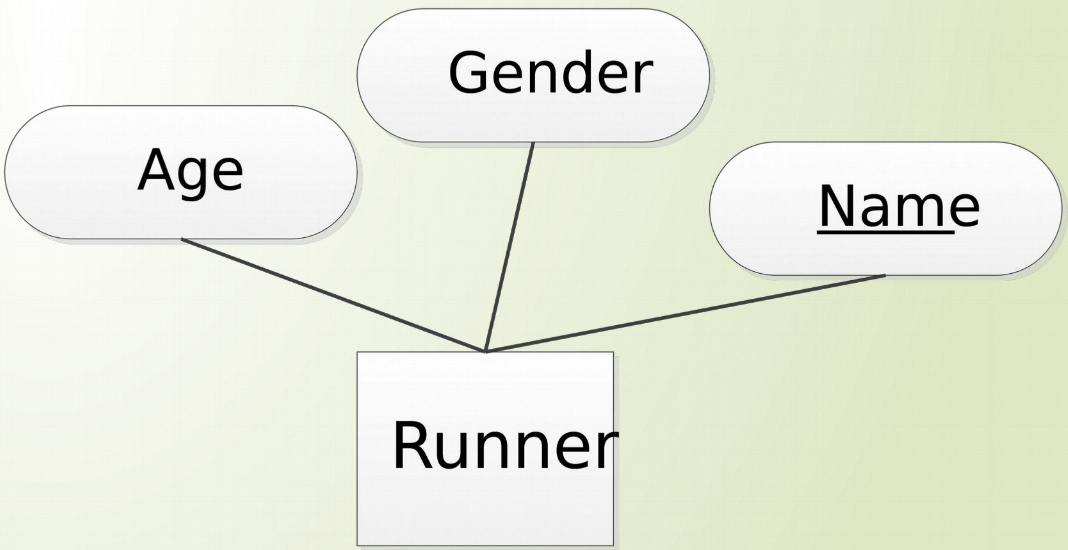
E/R Diagrams to Relations

- Every entity set and relationship becomes a **relation** (table).

- Convert each entity set to a relation
  - Determine keys
- Convert each relationship to a relation
  - Using entity set keys

From Entity Set to Relation

- Entity set attributes become relational attributes



Relation Keys

- Select the designated key of the corresponding entity set.
  - Or introduce a surrogate key.
- Underline the attribute(s) forming the key.
  - Ex: Runner(<u>name</u>, gender, age)
- No two tuples in a relation instance will have the same values for all of its key attributes.

Primary key

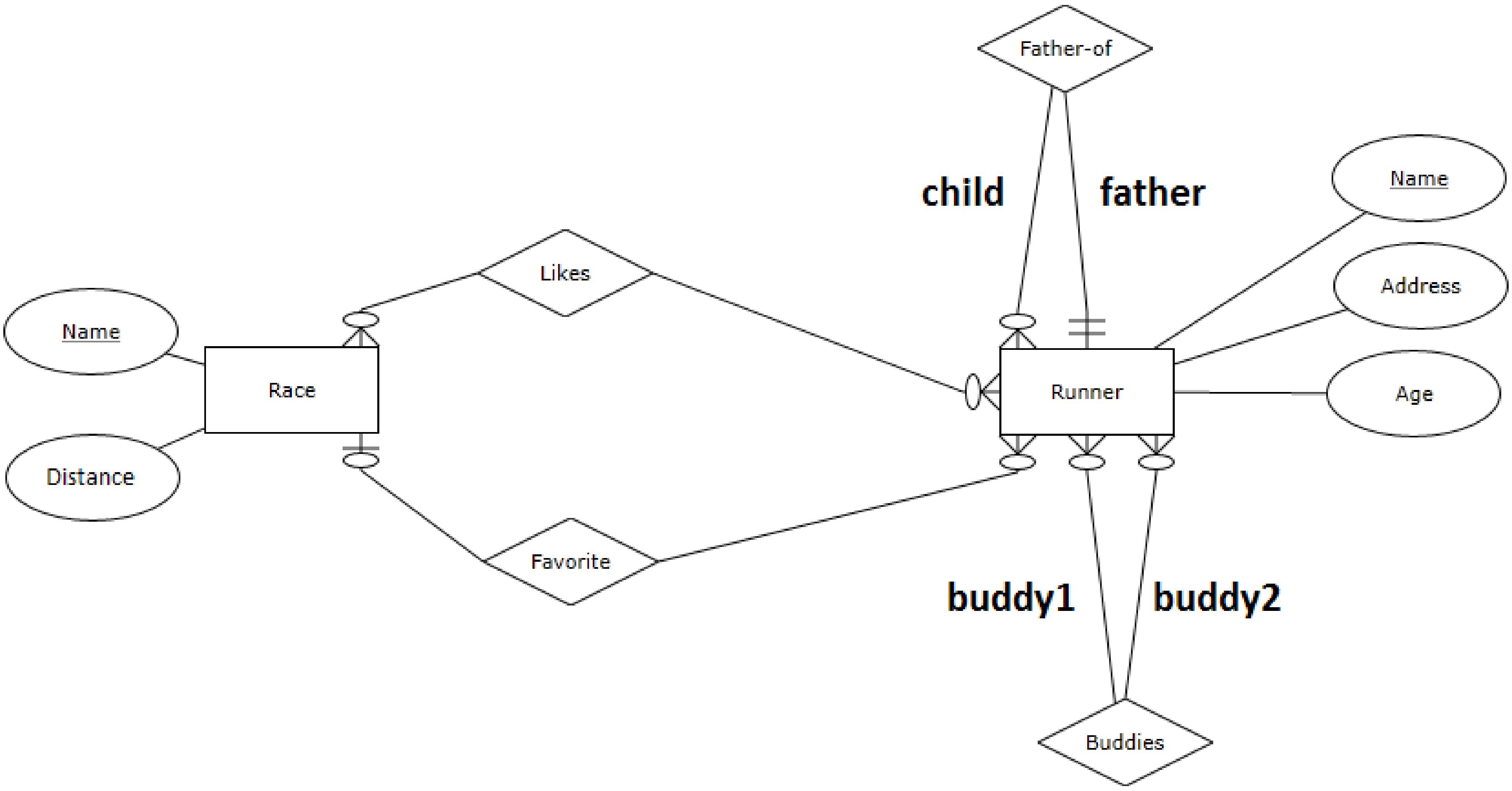
- **Primary key -** column (or a set of columns) whose value is unique for each row
  - Each relation must have a primary key
  - The name of the primary key column is underlined in order to distinguish it from the other columns in the relation

E/R Relationship to Relation

- Has attributes for key attributes of each entity set that participates in the relationship
  - And relationship attributes (if any)

- Naming conventions
  - Renaming attributes is ok; necessary if entity sets have attributes with same names

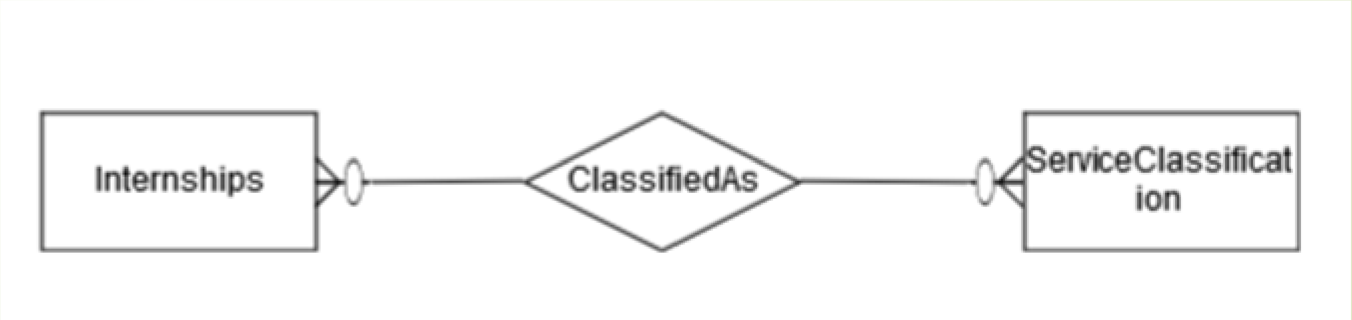
Example


Father-of
child
father
Name
Likes
Address
Name
Race
Runner
Age
Distance
Favorite
buddy1
buddy2
Buddies

Combining relations

- Common case: relation for entity set x plus the relation for some many-to-one relationship from x to another entity set.
- Example:
  - Runner(<u>name</u>, age) with
  - Favorite(<u>runner</u>, race) results in:
  - Runner1(<u>name</u>, age, favRace)
- **Many-to-one** multiplicity of the relationship (to be combined) is a necessary condition.

Real-World Example



Not combining relations

- (Bad) Example:
  - Runner(<u>name</u>, age) with
  - Likes(<u>runner</u>, <u>race</u>) results in
  - **RunnerCombine(<u>name</u>, age, likedRa**

name	age	likedRace
Kathryn	25	Cherry Blossom 10 mile
Kathryn	25	Arkansas Traveler 100

- The Runner’s age is repeated!

From Relation to Entity Set

- Can we reverse engineer: relations to ERD?

Comedian(<u>Name</u>, ComedyType, Email)

Venue(<u>Name</u>, Address, Website)

Performance(<u>ComedianName</u>, <u>VenueName</u>, Date, StartTime, EndTime)

Weak entity set to relations

- Relation for a weak entity set includes the full key as well as its own attributes.
- A supporting relationship (double diamond) yields a relation that is redundant and should not be included in the schema.

Foreign key

- **Foreign key -** *column in a relation that refers to a primary key column in another (referred) relation*
  - A mechanism that is used to depict relationships in the relational database model
  - For every occurrence of a foreign key, the relational schema contains a line pointing *from* the *foreign key* to the *corresponding primary key*

Functional Dependencies

- **Functional dependency -** occurs when the value of one (or more) column(s) in each record of a relation uniquely determines the value of another column in that same record of the relation
- For example:

A → B

ClientID → ClientName

RunnerName → RunnerAge

FD Example

- Runner(name, age, likedRace, raceDistance, favRace)

name	age	likedRace	distance	favRace
Kathryn	25	Cinco de Miler	5	Cherry Blossom 10
Kathryn	25	Burgers & Beer	13	Cherry Blossom 10
Zach	34	Cinco de Miler	5	Chicago Marathon
Nirajan	32	ChiTown Half	13.2	Paris Semi-marathon
Zach	34	Climb Hot Springs	7	Chicago Marathon

Properties of FDs

- Keys determine all attributes
- When FDs are not of the form “key determines other attribute(s)” there is typically an attempt to put too much into a single relation
  - Redundancy problems should be addressed and fixed.

Key and Superkey

- **Superkey**: Determines unique row
- **Key**: MINIMAL column(s) needed to determine unique row
  - If no subset of a superkey is also a superkey then it is a key

Key and Superkey

- Example: determine all keys and superkeys for Runner:
  - Runner(name, age, likedRace, distance, favRace)

Getting Keys and FDs

- Postulate a key directly
  - Surrogate keys
- Assert FDs and derive a key
- FDs come from keyness, many-one, one-one relationships

Inferring FDs (transitive rule)

- Given a list of FDs does another FD hold in a relation?
- E.g. $A \rightarrow B$, $B \rightarrow C$, then surely $A \rightarrow C$
- In general, you can check if FD $Y \rightarrow Z$ holds by:
  - Assume two tuples agree on Y attributes
  - Use existing FDs to infer other attributes on which they must agree
  - If Z is among them, then $Y \rightarrow Z$ holds.

Closure of attributes

- A test for FDs: attribute closure.

- Compute the closure of Y, designated as Y+ (all A such that Y → A)
- Starting point: Y+ = Y
- Induction: Look for FDs X → A such that X is a subset of Y+ and add A to Y+
- Stop when Y+ cannot be changed.

Closure example

- Relation R(A,B,C,D)
- Given FDS: A→B, B→CD
- Find the closure of all sets of attributes.

Closure example 2

- Relation R(A,B,C,D,E,F)
- $AB \rightarrow D$, $BD \rightarrow E$, $AE \rightarrow F$
- Find the closure of all sets of attributes.

Given vs Implied FDs

- Start with the FDs given by your client (or determined from E/R diagram or existing database)
- Other FDs may follow logically from the given FDs: these are **implied** FDs.
- In terms of design process there is no difference between the two types.

AB→D, BD→E, AE→F

means

AB→DE, AB→DEF

Non-Generic

- From previous example:

- WineName(A)/WineType(B) → Vineyard(D)
- WineType(B)/Vineyard(D) → GrapeUsed(E)
- WineName(A)/GrapeUsed(E) → TaninCount(F)

Finding all implied FDs

- For each set of attributes X, compute X+
- Add X→A for each A in X+
- Ignore “obvious” dependencies that follow from others:
  - Trivial FDs:
    - Right side is a subset of the left (XY→X)
    - Ignore XY→A, if X→A holds

Finding all implied FDs

- Motivation:
  - Determine the FDs and break up the relation into several smaller relations via normalization.
  - Determine what FDs hold for the smaller relations.

- Example:
  - R(A,B,C,D)
  - FDs AB✉C, C✉D
  - Decompose R into R1(A,B,C) and R2(C,D)

Projecting FDs

- Relation R is **decomposed** into relations R1 and R2. Now, what are the functional dependencies for R1 and R2?

- Given all given and implied functional dependencies for relation R, choose the ones with attributes in R1 for R1, and the ones with attributes in R2 for R2.

Example

- FDs = A → B, C → D, AC → E
- Decompose using FDs

- Example:
- Runner(Name, Age, Race, RaceDistance, PlaceEarned)

Summary

- E/R Modeling
- Relational model
- Converting E/R to relations
- Functional Dependencies (FDs)

Next time

- Normalizing relations
  - Using functional dependencies (FDs) and decomposition

- Manipulating relations
  - Relational algebra
  - SQL!

Second Homework

- Details in Canvas (on Thursday)

- Second Gradiance problem set
- Create ERD/schema for proposed project
- Recommend using ERDPlus
- Post questions in Piazza


QUESTIONS?
COMMENTS?
CONCERNS?