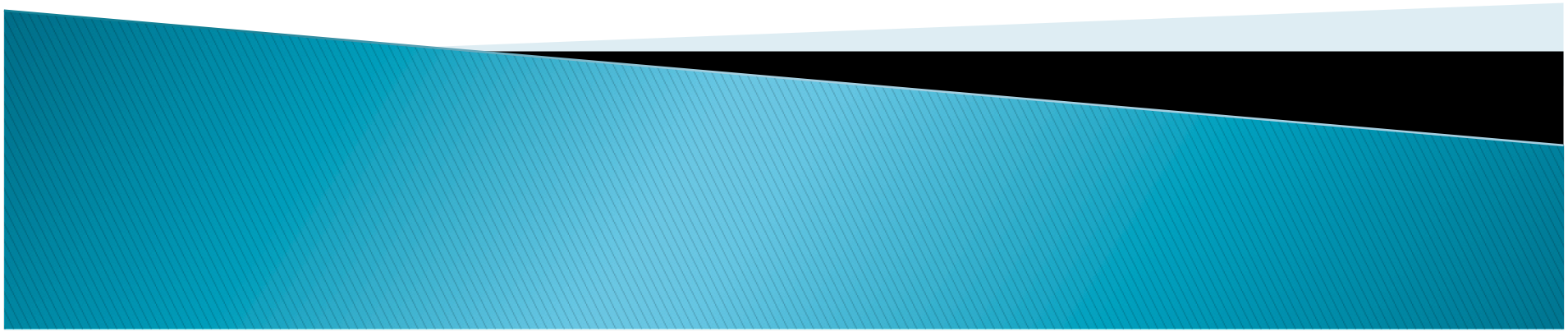
CMSC424: Normalization

Instructor: Amol Deshpande
amol@cs.umd.edu



Relational Database Design

- Where did we come up with the schema that we used ?
  - E.g. why not store the actor names with movies ?
- If from an E-R diagram, then:
  - Did we make the right decisions with the E-R diagram ?
- Goals:
  - Formal definition of what it means to be a “good” schema.
  - How to achieve it.

Movie(*title, year*, length, inColor, studioName, producerC#, starName)

Title	Year	Length	inColor	StudioName	prodC#	StarName
Star wars	1977	121	Yes	Fox	128	Hamill
Star wars	1977	121	Yes	Fox	128	Fisher
Star wars	1977	121	Yes	Fox	128	H. Ford
King Kong	2005	187	Yes	Universal	150	Watts
King Kong	1933	100	no	RKO	20	Fay

Issues:

1. Redundancy ➔ higher storage, inconsistencies ("anomalies")

   update anomalies, insertion anamolies

2. Need nulls

   Unable to represent some information without using nulls

   How to store movies w/o actors (pre-productions etc) ?

Key Problem:

There are *functional dependencies: A → B such that A is duplicated*

Definition: A → B: If two tuples agree on A, they agree on B

Therefore: If A is duplicated, B is duplicated

Movie(*title, year*, length, inColor, studioName, producerC#, starNames)

Title	Year	Length	inColor	StudioName	prodC#	StarNames
Star wars	1977	121	Yes	Fox	128	{Hamill, Fisher, H. ford}
King Kong	2005	187	Yes	Universal	150	Watts
King Kong	1933	100	no	RKO	20	Fay

Issues:

3. Avoid sets

- Hard to represent
- Hard to query

Smaller schemas always good ????

Split Studio(*name*, address, presC#) into:

Studio1 (name, presC#)

Name	presC#
Fox	101
Studio2	101
Universial	102

Studio2(name, address)???

Name	Address
Fox	Address1
Studio2	Address1
Universial	Address2

This process is also called *"decomposition"*

Issues:

4. Requires more joins (w/o any obvious benefits)
5. Hard to check for some dependencies

What if the "address" is actually the presC#'s address ?

No easy way to ensure that constraint (w/o a join).

Smaller schemas always good ????

Decompose StarsIn(movieTitle, movieYear, starName) into:

StarsIn1(movieTitle, movieYear)

movieTitle	movieYear
Star wars	1977
King Kong	1933
King Kong	2005

StarsIn2(movieTitle, starName) ???

movieTitle	starName
Star Wars	Hamill
King Kong	Watts
King Kong	Faye

Issues:

6. "joining" them back results in more tuples than what we started with

(King Kong, 1933, Watts) & (King Kong, 2005, Faye)

This is a "lossy" decomposition

We lost some constraints/information

The previous example was a "lossless" decomposition.

Desiredata

- No sets
- Correct and faithful to the original design
  - Avoid lossy decompositions
- As little redundancy as possible
  - To avoid potential anomalies
- No “inability to represent information”
  - Nulls shouldn’t be required to store information
- Dependency preservation
  - Should be possible to check for constraints

Not always possible.

We sometimes relax these for:

simpler schemas, and *fewer joins during queries.*

Approach

1. We will encode and list all our knowledge about the schema
   - Functional dependencies (FDs)

     SSN → name (means: SSN "implies" length)
   - If two tuples have the same "SSN", they must have the same "name"

     movietitle → length ???? Not true.
   - But, (movietitle, movieYear) → length --- True.
2. We will define a set of rules that the schema must follow to be considered good
   - "Normal forms": 1NF, 2NF, 3NF, BCNF, 4NF, …
   - A normal form specifies constraints on the schemas and FDs
3. If not in a "normal form", we modify the schema

Rest…

- See 424 Normalization Slides or a textbook