CS411
Database Systems

05: Relational Schema Design

Why Do We Learn This?

Because we want to do good job!

Motivation

- We have designed ER diagram, and translated it into a relational db schema R = set of R1, R2, ...
- Now what?
- We can do the following
  - specify all relevant constraints over R
  - implement R in SQL
  - start using it, making sure the constraints always remain valid
- However, R may not be well-designed, thus causing us a lot of problems

Q: This a good design?

Persons with several phones:

Address	SSN	Phone Number
~~10 Green~~	$\boxed{123\text{-}321\text{-}99}$	~~(201) 555-1234~~
~~10 Green~~	$\boxed{123\text{-}321\text{-}99}$	~~(206) 572-4312~~
431 Purple	909-438-44	(908) 464-0028
431 Purple	909-438-44	(212) 555-4000

?

Redundancy

V1, Alex

V3, Alex

Sally

Sally

P1

P2

P3

P4

* Waste space

* Update!

* delete

Potential Problems

- Redundancy
- Update anomalies
- Deletion anomalies

How do We Obtain a Good Design?

R

↓

Normalization

↓

R*

(normal form)

- Start with the original db schema R
- Transform it until we get a good design R*
- Desirable properties for R*
  - must preserve the information of R
  - must have minimal amount of redundancy
  - must be dependency-preserving
    - if R is associated with a set of constraints C, then it should be easy to also check C over R*
  - (must also give good query performance)

OK, But ...

- How do we recognize a good design R*?
- How do we transform R into R*?
- What we need is the "theory" of …

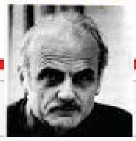
Normal Forms

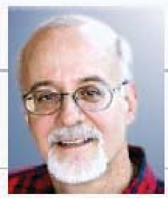
- DB gurus have developed many normal forms
- Most important ones
  - Boyce-Codd, 3rd, and 4th normal forms
- If R* is in one of these forms, then R* is guaranteed to achieve certain good properties
  - e.g., if R* is in Boyce-Codd NF, it is guaranteed to not have certain types of redundancy
- DB gurus have also developed algorithms to transform R into R* that is in some of these normal forms

Normal Forms (cont.)

- DB gurus have also discussed trade-offs among normal forms
- Thus, all we have to do is
  - learn these forms
  - transform R into R* in one of these forms
  - carefully evaluate the trade-offs
- Many of these normal forms are defined based on various constraints
  - functional dependencies and keys

Behind the Scene: Know whom we should blame?





Normal form	Defined by	Brief definition
First normal form (1NF)	Two versions: E.F. Codd (1970), C.J. Date (2003)[12]	Table faithfully represents a relation and has no "repeating groups"
Second normal form (2NF)	E.F. Codd (1971)[13]	No non-prime attribute in the table is functionally dependent on a part (proper subset) of a candidate key
Third normal form (3NF)	E.F. Codd (1971)[14]; see also Carlo Zaniolo's equivalent but differently-expressed definition (1982)[15]	Every non-prime attribute is non-transitively dependent on every key of the table
Boyce-Codd normal form (BCNF)	Raymond F. Boyce and E.F. Codd (1974)[16]	Every non-trivial functional dependency in the table is a dependency on a superkey
Fourth normal form (4NF)	Ronald Fagin (1977)[17]	Every non-trivial multivalued dependency in the table is a dependency on a superkey
Fifth normal form (5NF)	Ronald Fagin (1979)[18]	Every non-trivial join dependency in the table is implied by the superkeys of the table
Domain/key normal form (DKNF)	Ronald Fagin (1981)[19]	Every constraint on the table is a logical consequence of the table's domain constraints and key constraints
Sixth normal form (6NF)	Chris Date, Hugh Darwen, and Nikos Lorentzos (2002)[20]	Table features no non-trivial join dependencies at all (with reference to generalized join operator)

Our Attack Plan

- Motivation
- Functional dependencies & keys
- Reasoning with FDs and keys
- Desirable properties of schema refinement
- Various normal forms and the trade-offs
  – BCNF, 3rd normal form, 4th normal form, etc.
- Putting all together: how to design DB schema

Functional Dependencies and Keys

Better Designs Exist

Break the relation into two:

	SSN	Address
Alex	123-321-99	10 Green
Sally	909-438-44	431 Purple

Identify

?

Address ✗

SSN	Phone Number
123-321-99	(201) 555-1234
123-321-99	(206) 572-4312
909-438-44	(908) 464-0028
909-438-44	(212) 555-4000

A1
A2
A2
A2

Functional Dependencies

- A form of constraint (hence, part of the schema)
- Finding them is part of the database design
- Used heavily in schema refinement

Definition:

Name Address

If two tuples agree on the attributes

T1 Alex ...

Name $A_1, A_2, \ldots A_n$

T2 Alex ...

then they must also agree on the attributes

Address ~~B_1~~, $B_2, \ldots B_m$

Formally: $A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$