Database Normalization

CPS352: Database Systems

Simon Miner
Gordon College
Last Revised: 2/18/15

Agenda

- Check-in
- Functional Dependencies (continued)
- Team exercise
- Database Normalization

Check-in

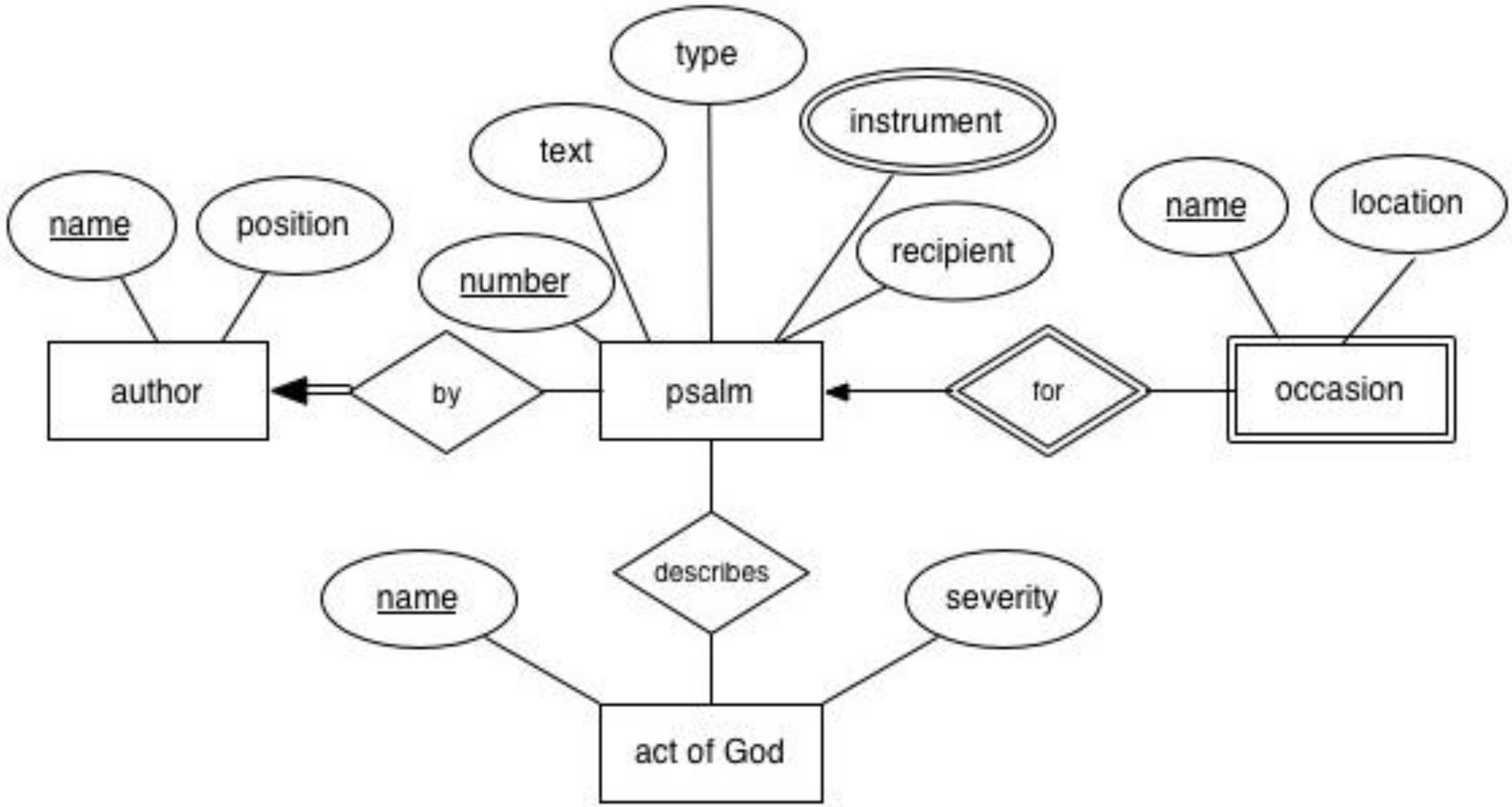
Datatabases

...of the Bible

Depending on the Psalms

Psalm 46 (NIV)

Psalm ERD



type
instrument
text
name
position
location
recipient
number
author
by
psalm
for
occasion
describes
name
severity
act of God

Database Normalization

The (Evolving) Art of Database Design

- Goals
  - Avoid redundancies and the resulting from insert, update, and delete anomalies by decomposing schemes as needed
  - Ensure that all decompositions are lossless-join
  - Ensure that all decompositions are dependency preserving
- Sometimes you cannot have all three
  - Allow for redundancy to preserve dependencies
  - Or give up dependency preservation to eliminate redundancy
  - **Never** give up lossless-join as doing so would remove the ability to connect tuples in different relations
- Database *normal forms* help eliminate redundancy and anomalies
  - Specify a set of decomposition rules to convert a database that is not in a given normal form into one that is

First Normal Form (1NF)

- A relation scheme R is in 1NF if the domains of all attributes in R are atomic
  - Single and non-composite
  - Guarantees that each non-key attribute in R is functionally dependent on the primary key

Second Normal Form (2NF)

- A 1NF relationship scheme R is in 2NF if each non-key attribute is fully functionally dependent on each candidate key
  - Functionally dependent on the whole key, not just part of it
    - This restriction does not apply to key attributes
  - Avoids redundancy of information which is dependent on part of the primary key
- Any non-2NF scheme can be decomposed into 2NF schemes by factoring out
  - The non-key attributes dependent on a portion of a candidate key
  - The portion of the candidate key these attributes depend on
- Any 1NF scheme without a composite primary key is in 2NF

Third Normal Form (3NF)

- A 2NF relation scheme R is in 3NF if no non-key attribute of R is transitively dependent on a candidate key through some other non-key attribute(s)
  - This restriction does not apply to key attributes
  - Transitive dependencies on a candidate key lead to insert, update, and delete anomalies
- Any non-3NF scheme can be decomposed into 3NF schemes by factoring out
  - The transitively dependent attributes
  - The “transitional” attributes which connect these to the candidate key
- Any non-3NF relation can be decomposed into 3NF in a lossless-join and dependency preserving manner

3NF Decomposition Algorithm

Let F_c be a canonical cover for *F;*
$i := 0;$
for each functional dependency $\alpha \rightarrow \beta$ in F_c **do**
  if none of the schemas R_j, $1 \leq j \leq i$ contains $\alpha\ \beta$
    then begin
      $i := i + 1;$
      $R_i := \alpha\ \beta$
    end
if none of the schemas R_j, $1 \leq j \leq i$ contains a candidate key for *R*
  then begin
    $i := i + 1;$
    $R_i :=$ any candidate key for *R;*
  end
/* Optionally, remove redundant relations */

repeat
if any schema R_j is contained in another schema R_k
  then /* delete R_j */
    $R_j = R_i;$
    $i=i-1;$
return $(R_1, R_2, \ldots, R_i)$

Boyce-Codd Normal Form (BCNF)

- 3NF did not take multiple candidate keys into account
  - BCNF developed to address this
- A normalized relation is in BCNF if every FD satisfied by R is of the form $A \rightarrow B$, where A is a superkey
  - BCNF is a stronger 3NF
  - Every BCNF schema is also in 3NF
  - Not every 3NF schema is in BCNF
- Some 3NF schemas cannot be decomposed into BCNF in a lossless-join and dependency preserving manner
- BCNF does not build on other normal forms

BCNF Decomposition Algorithm

```
result := {R };
done := false;
compute F+;
while (not done) do
  if (there is a schema Ri in result  that is not in BCNF)
     then begin
           let α → β  be a nontrivial functional dependency that
                holds on Ri  such that α → Ri is not in F+,
                and α ∩ β  = ∅;
            result := (result – Ri) ∪ (Ri – β) ∪ (α, β );
           end
     else done := true;
```

Note: each R_i is in BCNF, and decomposition is lossless-join.

Multivalued Dependencies (MVDs)

- A set of attributes A *multi-determines* a set of attributes B if
  - In any relation including attributes A and B
  - For any given value of A there is a (non-empty) set of values for B
  - Such that we expect to see all of those B values (and no others) associated with the given A
  - Along with remaining attribute values
  - The number of B values associated with a given A value may vary between A values.

Formal Definition of Multivalued Dependency

- Let R be a relation schema and let $\alpha \subseteq R$ and $\beta \subseteq R$. The **multivalued dependency**

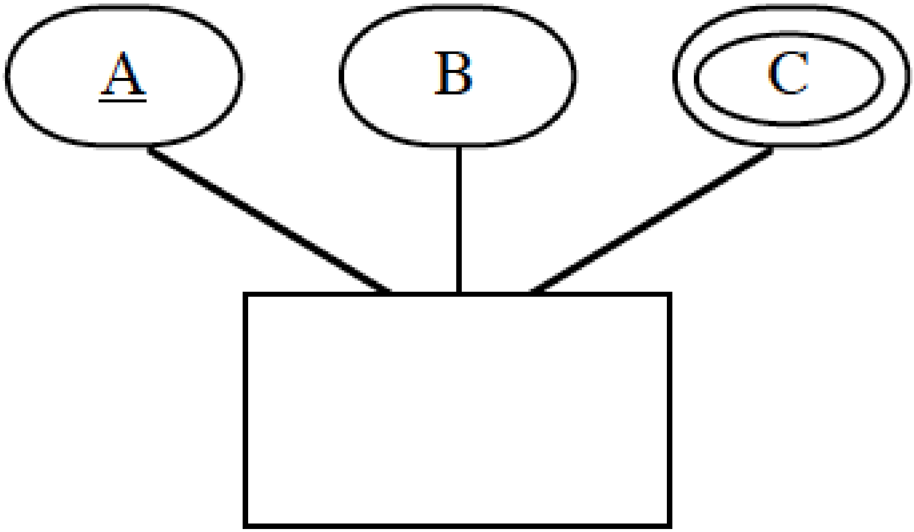
$$\alpha \rightarrow\rightarrow \beta$$

holds on R if in any legal relation $r(R)$, for all pairs for tuples t_1 and t_2 in r such that $t_1[\alpha] = t_2[\alpha]$, there exist tuples t_3 and t_4 in r such that:

$$\begin{aligned} t_1[\alpha] = t_2[\alpha] = t_3[\alpha] &= t_4[\alpha] \\ t_3[\beta] &= t_1[\beta] \\ t_3[R - \beta] &= t_2[R - \beta] \\ t_4[\beta] &= t_2[\beta] \\ t_4[R - \beta] &= t_1[R - \beta] \end{aligned}$$

MVDs and E-R Diagrams

- MVDs correspond to multi-valued attributes



$$A \rightarrow B$$
$$A \twoheadrightarrow C$$

Properties of MVDs

- MVDs require the addition of certain tuples
  - Example: copies of a book with multiple authors
  - Opposite to FDs which prohibit certain tuples
- If $A \rightarrow B$, then $A \rightarrow\rightarrow B$
  - FDs are a special case of MVDs
- An MVD is trivial if either of the following is true
  - Its right-hand side is a subset of its left-hand side (just like FDs)
  - The union of its left- and right-hand sides is the entire scheme
- The closure D+ of D is the set of all FDs and MVDs implied by D
  - D+ can be computed from the formal definitions of FD and MVD
  - Additional rules of inference (see Appendix C of *Database Systems Concepts*)

Fourth Normal Form (4NF)

- A relation schema *R* is in **4NF** for all MVDs in D^+ of the form $\alpha \rightarrow\rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
  - $\alpha \rightarrow\rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
  - α is a superkey for schema *R* (in which case it is an FD)
- If a relation is in 4NF it is in BCNF
- 4NF avoids redundancies introduced by MVDs

4NF Decomposition Algorithm

result: = {*R*};
done := false;
compute D^+;
Let D_i denote the restriction of D^+ to R_i

while (**not** *done*)
if (there is a schema $\mathbf{R}_i$ in *result* that is not in 4NF) **then**
begin

let $\alpha \rightarrow\rightarrow \beta$ be a nontrivial multivalued dependency that holds
on R_i such that $\alpha \rightarrow R_i$ is not in D_i, and $\alpha \cap \beta = \phi$;
result := (*result* - R_i) $\cup$ (R_i - β) $\cup$ (α, β);
end
else *done*:= true;

Note: each R_i is in 4NF, and decomposition is lossless-join

Database Design Guidelines

- Use the highest normal form possible
  - 4NF unless it is not dependency preserving
  - BCNF unless (in rare cases) it is not dependency preserving
  - 3NF otherwise – never need to compromise beyond this
  - Lower normal forms may be useful for efficiency purposes
- Use good keys
  - Every attribute should depend on the key, the whole key, and nothing but the key (BCNF)
  - Avoid composite keys (automatic 2NF)
    - Generate a unique single-attribute key if needed
- Factor out transitive dependencies ("sub-relations") into their own schemes (3NF
- Isolate MVDs to their own schemes (4NF)

Approaches to Database Design

- Start with a universal relation and decompose it
  - The approach taken in this lecture
- Start with an E-R diagram
  - Modify it while you normalize it
  - Normalize it when converting it to a relational schema