Relational Database Design

Jan Chomicki
University at Buffalo

Outline

“Good” and “bad” database schemas

“Good” and “bad” database schemas

“Bad” schema

- Repetition of information. Leads to redundancies, potential inconsistencies, and update anomalies.
- Inability to represent information. Leads to anomalies in insertion and deletion.

“Good” and “bad” database schemas

“Bad” schema

- Repetition of information. Leads to redundancies, potential inconsistencies, and update anomalies.
- Inability to represent information. Leads to anomalies in insertion and deletion.

“Good” schema

- relation schemas in normal form (redundancy- and anomaly-free): BCNF, 3NF.

"Good" and "bad" database schemas

"Bad" schema

- Repetition of information. Leads to redundancies, potential inconsistencies, and update anomalies.
- Inability to represent information. Leads to anomalies in insertion and deletion.

"Good" schema

- relation schemas in normal form (redundancy- and anomaly-free): BCNF, 3NF.

Schema decomposition

- improving a bad schema
- desirable properties:
  - lossless join
  - dependency preservation

Integrity constraints

Integrity constraints

Functional dependencies

- key constraints cannot express uniqueness properties holding in a proper subset of all attributes
- key constraints need to be generalized to functional dependencies

Integrity constraints

Functional dependencies

- key constraints cannot express uniqueness properties holding in a proper subset of all attributes
- key constraints need to be generalized to functional dependencies

Other constraints

- not relevant for decomposition

Functional dependencies (FDs)

Functional dependencies (FDs)

Notation

- Relation schema $R(A_1, \ldots, A_n)$
- r is an instance of R
- Sets of attributes of R: $X, Y, Z, \ldots \subseteq \{A_1, \ldots, A_n\}$
- $A_1 \cdots A_n$ instead of $\{A_1, \ldots, A_n\}$.
- XY instead of $X \cup Y$.

Functional dependencies (FDs)

Notation

- Relation schema $R(A_1, \ldots, A_n)$
- r is an instance of R
- Sets of attributes of R: $X, Y, Z, \ldots \subseteq \{A_1, \ldots, A_n\}$
- $A_1 \cdots A_n$ instead of $\{A_1, \ldots, A_n\}$.
- XY instead of $X \cup Y$.

Functional dependency

- syntax: $X \to Y$
- semantics: r satisfies $X \to Y$ if for all tuples $t_1, t_2 \in r$:

$$\textit{if } t_1[X] = t_2[X], \textit{ then also } t_1[Y] = t_2[Y].$$

Dependency implication

Dependency implication

Implication

A set of FDs F implies an FD $X \to Y$, if every relation instance that satisfies all the dependencies in F, also satisfies $X \to Y$.

Notation

$F \models X \to Y$ (F implies $X \to Y$).

Dependency implication

Implication

A set of FDs F implies an FD $X \to Y$, if every relation instance that satisfies all the dependencies in F, also satisfies $X \to Y$.

Notation

$F \models X \to Y$ (F implies $X \to Y$).

Closure of a dependency set F

The set of dependencies implied by F.

Notation

$F^+ = \{X \to Y : F \models X \to Y\}$.

Keys

Keys

Key

$X \subseteq \{A_1, \ldots, A_n\}$ is a key of R if:

1. the dependency $X \rightarrow A_1 \cdots A_n$ is in F^+.
2. for all proper subsets Y of X, the dependency $Y \rightarrow A_1 \cdots A_n$ is not in F^+.

Keys

Key

$X \subseteq \{A_1, \ldots, A_n\}$ is a key of R if:

1. the dependency $X \to A_1 \cdots A_n$ is in F^+.
2. for all proper subsets Y of X, the dependency $Y \to A_1 \cdots A_n$ is not in F^+.

Related notions

- *superkey:* superset of a key.
- *primary key:* one designated key.
- *candidate key:* one of the keys.

Inference of functional dependencies

Inference of functional dependencies

Dependency inference

How to tell whether $X \rightarrow Y \in F^{+}$?

Inference of functional dependencies

Dependency inference

How to tell whether $X \to Y \in F^+$?

Inference rules (Armstrong axioms)

1. reflexivity: infer $X \to Y$ if $Y \subseteq X \subseteq attrs(R)$ (*trivial* dependency)
2. augmentation: From $X \to Y$ infer $XZ \to YZ$ if $Z \subseteq attrs(R)$
3. transitivity: From $X \to Y$ and $Y \to Z$, infer $X \to Z$.

Properties of axioms

Properties of axioms

Armstrong axioms are:

- sound: if $X \to Y$ is derived from F, then $X \to Y \in F^+$.
- complete: if $X \to Y \in F^+$, then $X \to Y$ is derived from F.

Properties of axioms

Armstrong axioms are:

- sound: if $X \to Y$ is derived from F, then $X \to Y \in F^+$.
- complete: if $X \to Y \in F^+$, then $X \to Y$ is derived from F.

Additional (implied) inference rules

4. union: from $X \to Y$ and $X \to Z$, infer $X \to YZ$
5. decomposition: from $X \to Y$ infer $X \to Z$, if $Z \subseteq Y$

Boyce-Codd Normal Form (BCNF) and 3NF

Boyce-Codd Normal Form (BCNF) and 3NF

BCNF

A schema R is in BCNF if for every nontrivial FD $X \rightarrow A \in F$, X contains a key of R.

Boyce-Codd Normal Form (BCNF) and 3NF

BCNF

A schema R is in BCNF if for every nontrivial FD $X \to A \in F$, X contains a key of R.

Each instance of a relation schema which is in BCNF does not contain a redundancy (that can be detected using FDs alone).

Boyce-Codd Normal Form (BCNF) and 3NF

BCNF

A schema R is in BCNF if for every nontrivial FD $X \rightarrow A \in F$, X contains a key of R.

Each instance of a relation schema which is in BCNF does not contain a redundancy (that can be detected using FDs alone).

3NF

R is in 3NF if for every nontrivial FD $X \rightarrow A \in F$:

- X contains a key of R, or
- A is part of some key of R.

Boyce-Codd Normal Form (BCNF) and 3NF

BCNF

A schema R is in BCNF if for every nontrivial FD $X \to A \in F$, X contains a key of R.

Each instance of a relation schema which is in BCNF does not contain a redundancy (that can be detected using FDs alone).

3NF

R is in 3NF if for every nontrivial FD $X \to A \in F$:

- X contains a key of R, or
- A is part of some key of R.

BCNF vs. 3NF

- if R is in BCNF, it is also in 3NF
- there are relations that are in 3NF but not in BCNF.

Decompositions

Decompositions

Decomposition

Replacement of a relation schema R by two relation schema R_1 and R_2 such that $R = R_1 \cup R_2$.

Decompositions

Decomposition

Replacement of a relation schema R by two relation schema R_1 and R_2 such that $R = R_1 \cup R_2$.

Lossless-join decomposition

(R_1, R_2) is a lossless-join decomposition of R with respect to a set of FDs F if for every instance r of R that satisfies F:

$$\pi_{R_1}(r) \bowtie \pi_{R_2}(r) = r.$$

Decompositions

Decomposition

Replacement of a relation schema R by two relation schema R_1 and R_2 such that $R = R_1 \cup R_2$.

Lossless-join decomposition

(R_1, R_2) is a lossless-join decomposition of R with respect to a set of FDs F if for every instance r of R that satisfies F:

$$\pi_{R_1}(r) \bowtie \pi_{R_2}(r) = r.$$

A simple criterion for checking whether a decomposition (R_1, R_2) is lossless-join:

- $R_1 \cap R_2 \rightarrow R_1 \in F^+$, or
- $R_1 \cap R_2 \rightarrow R_2 \in F^+$.

Decompositions

Decomposition

Replacement of a relation schema R by two relation schema R_1 and R_2 such that $R = R_1 \cup R_2$.

Lossless-join decomposition

(R_1, R_2) is a lossless-join decomposition of R with respect to a set of FDs F if for every instance r of R that satisfies F:

$$\pi_{R_1}(r) \bowtie \pi_{R_2}(r) = r.$$

A simple criterion for checking whether a decomposition (R_1, R_2) is lossless-join:

- $R_1 \cap R_2 \to R_1 \in F^+$, or
- $R_1 \cap R_2 \to R_2 \in F^+$.

Decomposition into more than two schemas

- generalized definition
- more complex losslessness test

Dependency preservation

Dependency preservation

Dependencies associated with relation schema R_1 and R_2 in a decomposition (R_1, R_2):

$$F_{R_1} = \{X \rightarrow Y | X \rightarrow Y \in F^+ \wedge XY \subseteq R_1\}$$

$$F_{R_2} = \{X \rightarrow Y | X \rightarrow Y \in F^+ \wedge XY \subseteq R_2\}.$$

Dependency preservation

Dependencies associated with relation schema R_1 and R_2 in a decomposition (R_1, R_2):

$$F_{R_1} = \{X \rightarrow Y | X \rightarrow Y \in F^+ \wedge XY \subseteq R_1\}$$

$$F_{R_2} = \{X \rightarrow Y | X \rightarrow Y \in F^+ \wedge XY \subseteq R_2\}.$$

(R_1, R_2) preserves a dependency f iff $f \in (F_{R_1} \cup F_{R_2})^+$.

Decomposition into BCNF

Decomposition into BCNF

Algorithm: decomposition of schema R

1. For some nontrivial nonkey dependency $X \rightarrow A$ in F^+:
   - create a relation schema R_1 with the set of attributes XA and FDs F_{R_1}.
   - create a relation schema R_2 with the set of attributes $R - \{A\}$ and FDs F_{R_2}.
2. Decompose further the resulting schemas which are not in BCNF.

Decomposition into BCNF

Algorithm: decomposition of schema R

1. For some nontrivial nonkey dependency $X \to A$ in F^+:
    - create a relation schema R_1 with the set of attributes XA and FDs F_{R_1}.
    - create a relation schema R_2 with the set of attributes $R - \{A\}$ and FDs F_{R_2}.
2. Decompose further the resulting schemas which are not in BCNF.

This algorithm produces a lossless-join decomposition into BCNF which does not have to preserve dependencies.

Decomposition (synthesis) into 3NF

Decomposition (synthesis) into 3NF

Minimal basis F' for F

- set of FDs equivalent to F ($F^+ = (F')^+$),
- all FDs in F' are of the form $X \rightarrow A$ where A is a single attribute,
- further simplification by removing dependencies or attributes from dependencies in F' yields a set of FDs inequivalent to F.

Decomposition (synthesis) into 3NF

Minimal basis F' for F

- set of FDs equivalent to F ($F^+ = (F')^+$),
- all FDs in F' are of the form $X \to A$ where A is a single attribute,
- further simplification by removing dependencies or attributes from dependencies in F' yields a set of FDs inequivalent to F.

Algorithm: 3NF synthesis

1. Create a minimal basis F'.
2. Create a relation with attributes XA for every dependency $X \to A \in F'$.
3. Create a relation X for some key X of R.
4. Remove redundancies.

Decomposition (synthesis) into 3NF

Minimal basis F' for F

- set of FDs equivalent to F ($F^+ = (F')^+$),
- all FDs in F' are of the form $X \to A$ where A is a single attribute,
- further simplification by removing dependencies or attributes from dependencies in F' yields a set of FDs inequivalent to F.

Algorithm: 3NF synthesis

1. Create a minimal basis F'.
2. Create a relation with attributes XA for every dependency $X \to A \in F'$.
3. Create a relation X for some key X of R.
4. Remove redundancies.

This algorithm produces a lossless-join decomposition into 3NF which preserves dependencies.

Multivalued dependencies (MVDs)

Multivalued dependencies (MVDs)

Notation

- Relation schema $R(A_1, \ldots, A_n)$.
- r is an instance of R
- Sets of attributes: $X, Y, Z, \ldots \subseteq \{A_1, \ldots, A_n\}$.

Multivalued dependencies (MVDs)

Notation

- Relation schema $R(A_1, \ldots, A_n)$.
- r is an instance of R
- Sets of attributes: $X, Y, Z, \ldots \subseteq \{A_1, \ldots, A_n\}$.

Multivalued dependency

- syntax: a pair $X \rightarrow\rightarrow Y$.
- semantics: r satisfies $X \rightarrow\rightarrow Y$ if for all tuples $t_1, t_2 \in r$:

  if $t_1[X] = t_2[X]$, *then there is a tuple* $t_3 \in r$ *such that* $t_3[XY] = t_1[XY]$ *and* $t_3[Z] = t_2[Z]$,

  where $Z = \{A_1, \ldots, A_n\} - XY$.

Multivalued dependencies (MVDs)

Notation

- Relation schema $R(A_1, \ldots, A_n)$.
- r is an instance of R
- Sets of attributes: $X, Y, Z, \ldots \subseteq \{A_1, \ldots, A_n\}$.

Multivalued dependency

- syntax: a pair $X \rightarrow\rightarrow Y$.
- semantics: r satisfies $X \rightarrow\rightarrow Y$ if for all tuples $t_1, t_2 \in r$:

  if $t_1[X] = t_2[X]$, *then there is a tuple* $t_3 \in r$ *such that* $t_3[XY] = t_1[XY]$ *and* $t_3[Z] = t_2[Z]$,

  where $Z = \{A_1, \ldots, A_n\} - XY$.

Implication

Defined in the same way as for FDs.

Fourth Normal Form (4NF)

Fourth Normal Form (4NF)

F is the set of FDs and MVDs associated with a relation schema $R = \{A_1, \ldots, A_n\}$.

Fourth Normal Form (4NF)

F is the set of FDs and MVDs associated with a relation schema $R = \{A_1, \ldots, A_n\}$.

4NF

R is in 4NF if for every multivalued dependency $X \rightarrow\rightarrow Y$ entailed by F:

- $Y \subseteq X$ or $XY = \{A_1, \ldots, A_n\}$ (trivial MVD), or
- X contains a key of R.