Database Design Process

1. Functional Specifications
2. Database Specifications

Database Specifications

ER Modelling => Conceptual Schema Design
Relational Model => Mapping to DBMS Data Model
Normal Forms => Scheme Refinement
Tuning, index, selection, ... => Physical Design

Background

Example One

Original

Vendors

vno	vname	vaddress
v10	Bay	Southgate Mall, Edmonton
v20	Wall Mart	Calgary Trail, Edmonton
v30	Sears	Golden Mall, Calgary

Customers

account	cname	caddress	province	balance	crlimit
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000
a3	J. Miller	34-75 Ave, Edmonton	AB	500	2500
a4	D. Rafiei	111-22 Ave, Edmonton	AB	0	1200

Transactions

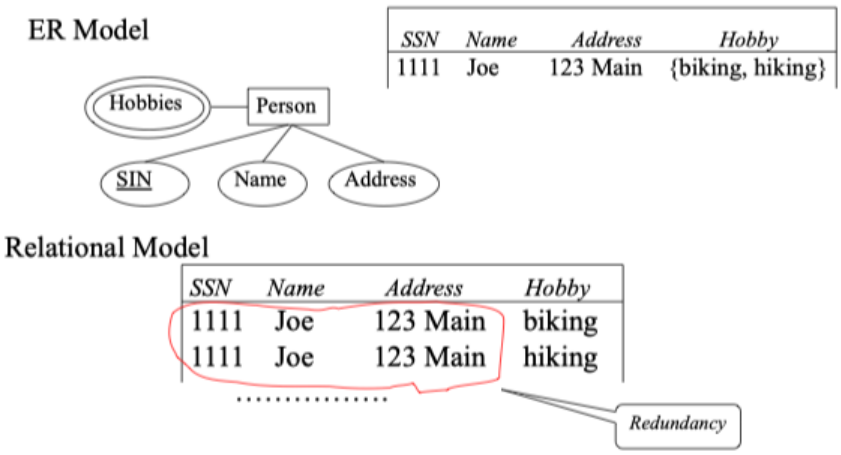
account	vno	amount
a1	v30	68.29
a2	v10	25.86
a1	v10	56.37
a3	v20	12.40
a2	v30	38.12
a3	v10	75.70
a1	v20	68.00
a2	v10	21.60
a1	v30	88.20
a2	v10	25.50

Alternative Schema

SuperRelation

account	cname	caddress	province	balance	crlimit	vno	vname	vaddress	amount
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v30	Sears	Golden Mall, Calgary	68.29
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	v10	Bay	Southgate Mall, Edmonton	25.86
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v10	Bay	Southgate Mall, Edmonton	56.37
a3	J. Miller	34-75 Ave, Edmonton	AB	500	2500	v20	Wall Mart	Calgary Trail, Edmonton	12.40
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	v30	Sears	Golden Mall, Calgary	38.12
a3	J. Miller	34-75 Ave, Edmonton	AB	500	2500	v10	Bay	Southgate Mall, Edmonton	75.70
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v20	Wall Mart	Calgary Trail, Edmonton	68.00
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	v10	Bay	Southgate Mall, Edmonton	21.60
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v30	Sears	Golden Mall, Calgary	88.20
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	V10	Bay	Southgate Mall, Edmonton	25.50
a4	D. Rafiei	111-22 Ave, Edmonton	AB	0	1200	NULL	NULL	NULL	NULL

Example 2



- There exists a redundancy

Example 3

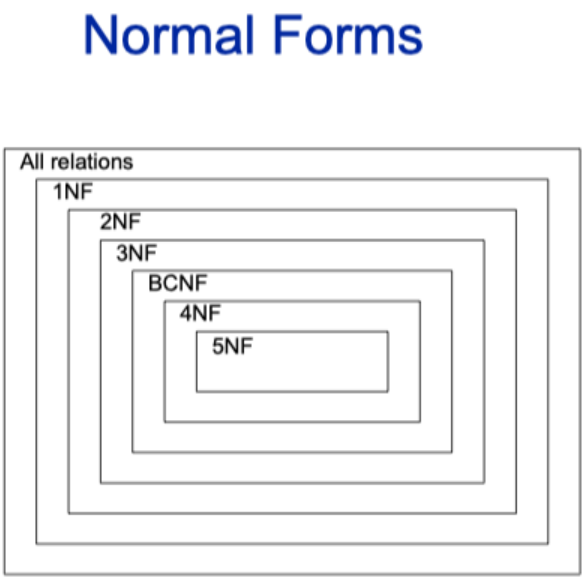
SSN	Name	Town	Zip
1234	Joe	Stony Brook	11790
4321	Mary	Stony Brook	11790
5454	Tom	Stony Brook	11790

- Another redundancy - all addresses in the same town have the same zip code

Guidelines

- Information repetition should be avoided
  - Anomalies: insertion, deletion, modification
- Avoid null values as much as possible
  - Difficulties with interpretation
    - dont know, dont care, known but unavailable, does not apply
  - Specification of joins

- General solution: **Normalization**



Schema Refinement

SuperRelation

account	cname	caddress	province	balance	crlimit	vno	vname	vaddress	amount
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v30	Sears	Golden Mall, Calgary	68.29
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	v10	Bay	Southgate Mall, Edmonton	25.86
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v10	Bay	Southgate Mall, Edmonton	56.37
a3	J. Miller	34-75 Ave, Edmonton	AB	500	2500	v20	Wall Mart	Calgary Trail, Edmonton	12.40
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	v30	Sears	Golden Mall, Calgary	38.12
a3	J. Miller	34-75 Ave, Edmonton	AB	500	2500	v10	Bay	Southgate Mall, Edmonton	75.70
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v20	Wall Mart	Calgary Trail, Edmonton	68.00
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	v10	Bay	Southgate Mall, Edmonton	21.60
a1	J. Doe	101-28 Ave, Edmonton	AB	800	1000	v30	Sears	Golden Mall, Calgary	88.20
a2	M. Smith	93-105 St, St. Albert	AB	1200	3000	V10	Bay	Southgate Mall, Edmonton	25.50
a4	D. Rafiei	111-22 Ave, Edmonton	AB	0	1200	NULL	NULL	NULL	NULL

Problems

Redundant storage

- some information stored repeatedly

Update Anomalies

- If one copy of such repeated data is updated, an inconsistency is created unless all copies are similarly updated

Insertion Anomalies

- It may not be possible to store certain information unless some other, unrelated, information is stored as well

Deletion Anomalies

- It may not be possible to delete certain information without losing some other, unrelated information as well

Problems w/ Decomposition

- Querying may require joining decomposed relations
  - the performance penalty of decomposing the relation may be unacceptable for too many joins
- In this case, we may choose to live with some problems of redundancy and not decompose the relation

Functional Dependencies

A functional dependency $X \rightarrow Y$ holds over relation R if whenever two tuples of R have the same X-value, they must also have the same Y-value

equivalent interpretation

$X \rightarrow Y$

- X determines Y
- Y is functionally dependent on X

1. this must hold over all instances
2. X can be a set of attributes

Example

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	e11

An Instance that Satisfies FD AB ➔ C

1. From the table FD is not the same as a key constraint - check first two tuples
2. The third and fourth tuples illustrate that if two tuples differ in either the A field or B field, they can differ in the C field without violating the FD

Implication for keys

1. K is a candidate key for R - means that **K$\rightarrow$ R**
2. A primary key constraint is a special case of an FD
   - X - attributes in the key
   - Y - the attributes in the relation
3. FD does not require that the set Y be minimal
4. The additional minimality condition must be met for X to be key

Example

$R(A, B, C, D, E)$
$\text{FDs} : \{\text{AB} \rightarrow \text{CDE}\}$

- AB is a superkey because $\text{AB} \rightarrow \text{ABCDE}$
  - AB determines all attributes in the relation
  - Since AB inherently includes A and B
- AB is a key because it is minimal
  - Neither A nor B alone can uniquely determine all the attributes of the relation

Implied FDs

Given some FDs we can usually infer additional FDs
Example: ssn $\rightarrow$ did, did $\rightarrow$ lot *implies* ssn $\rightarrow$ lot

Armstrong Axioms

Sound and complete inference rules for FDs

X, Y, Z are sets of attributes

Reflexivity: If $X \subset Y$ then $Y \to X$
Augmentation: If $X \to Y$, then $XZ \to YZ$ for any Z
Transitivity: If $X \to Y$ and $Y \to Z$, then $X \to Z$

Union and Decomposition

Additional rules that follow from AA
Union: If $X \to Y$, then $X \to Z$ then $X \to YZ$
Decomposition: If $X \to YZ$ then $X \to Y$ and $X \to Z$

Closure

Given a set F of FDs, the **closure of F** denoted by F+ is the set of all FS's logically implied by the FDs in F. F+ could be large

Definition

F : a given set of functional dependencies
$F+$: The set of all FDs that can be logically inferred from F
$X+$: The set of all attributes that can be functionally determined from X using F

Example 1

Question: Does $F = \{A \to B, B \to C, CD \to E\}$ imply $A \to E$?

- In other words, is $A \to E$ in the closure F^{+}?
- Equivalently, is E in A^{+} (closure of A)?

Steps to Compute A^{+}:

1. **Start with $A^{+} = \{A\}$:**
   - Initially, the closure of A only contains itself.
2. **Apply the Functional Dependency $A \to B$:**
   - Add B to A^{+}.
   - Now:

$$A^{+} = \{A, B\}$$

3. **Apply the Functional Dependency $B \to C$:**
   - Add C to A^{+}.
   - Now:

$$A^{+} = \{A, B, C\}$$

4. **Check for $CD \to E$:**
   - To use $CD \to E$, both C and D need to be in A^{+}.
   - However, $D \notin A^{+}$, so this functional dependency **cannot be applied**.

Conclusion

- Since $E \notin A^{+}$, the closure of A does **not** include E.
- Therefore, $A \to E$ **is not implied** by the given functional dependencies F.

Example 2

Relation: $R(A, B, C, D)$
Functional Dependencies: $F = \{A \to B, BC \to D\}$

Steps to Compute Closure and Key

1. **Given**:
   - $A^+ = AB$
   - $(AC)^+ = ABCD$
   - **Key**: AC is a candidate key.

Step-by-Step Explanation

1. **Start with** $(AC)^+ = \{A, C\}$
   - Because we begin with AC itself.
2. **Use Functional Dependencies to Expand** $(AC)^+$:
   - **Dependency**: $A \rightarrow B$
     - Since $A \in (AC)^+$, add B to the closure.
     - Now:

$$(AC)^+ = \{A, C, B\}$$

   - **Dependency**: $BC \rightarrow D$
     - Since B and C are both in $(AC)^+$, add D to the closure.
     - Now:

$$(AC)^+ = \{A, C, B, D\}$$

3. **Conclusion**:
   - $(AC)^+ = \{A, B, C, D\}$, which includes **all attributes** of the relation R.
   - Therefore, AC is a **minimal superkey** and hence a **candidate key**.

Roles of FDs in Detecting Redundancy

No FDs hold: means that there is no redundancy here

Consider

A relation R with 3 attributes ABC. If $A \rightarrow B$, then several tuples could have the same A value and if so , they all have the same B value

Normal Forms

Overview

A given relation schema represents a good design or needs to be decomposed it into smaller relations. If a relation is in a certain normal form

- It is know that certain kinds of problems are avoided/minimized
- This can be used to decide whether decomposing the relation will help

Types of Normal Forms based ON FDs

- 1NF, 2NF, 3NF, and Boyce-Codd Normal Form (BCNF)
- Every relation in BCNF are also in 3NF, every relation in 3NF is also in 2NF

BCNF: Boyce-Codd Normal Form (BCNF)

Definition

A relation R is in BCNF if for every non-trivial FD $X \rightarrow Y$ on R, where $X, Y \subset R$, X is a super key of R (i.e., X can determine all the attributes in the relation)

- $X \rightarrow Y$ is a trivial FD iff $Y \subset X$
  Thus. a schema is in BCNF if all its relations are in BCNF

Example 1

R(A,B,C,D,E,F), FDs={A->BC, D->EF}, is the relation in BCNF
FD1: A->BC

- this means that A determines BC but this alone cannot determine the rest of the attributes in R
- Thus A is not a super key - which violates BCNF
  FD2: D->EF
- D determines EF but not the rest of R
- Thus D is not a super key - which violates BCNF

Example 2

Given R(A,B,C,D,E,F) and the FDs {A->BC, D->AEF}

- are the relations R1(A, D, E,F) and R2(A, B, C) in BCNF? **yes**
- what about relations R2(B,C,D) and R4(A,B,D)? **no**

Why does a BCNF violation produce a "bad" relation?

Lets consider a real example, Loans(sid, name, address, isbn, title, author) with FDs={sid -> name, address, isbn -> author, title}.

1. sid only determines name and address
   - thus not a super key, violates BCNF
2. isbn only determines author and titile
   - thus not a superkey violates BCNF
   Both sid and icbn can form the candidate key
   Redundancy: Consider multiple loans by the same borrower (same sid):

- The name and address will be repeated for each loan
  Similarly, multiple loans of the same book (same isbn) will repeatedly store author and title

3NF: Third Normal Form

Definition

A relation R is in 3NF if for every trivial FD X->Y on R where X,Y is in R, C is a super key of R or Y is part of a key (prime attribute)

Example 1

R(A,B,C), FDs={AB->C, C->B}. Is R in 3NF? **yes**

- With AB->C, AB satisfies the first condition of BCNF
- With C->B, C is not a super key, but B is is a prime attributes - which satisfies second condition

Example 2

R(A,B,C), FD={A->, B->C}

1. With A->B and B->C implying A->C, A is a super key - satisfies the first condition
2. With B->C, B is not a super key, and C is not a prime attribute - this violates BCNF

2NF: Second Normal Form

A table is in 2NF if:

1. It is in 1NF
2. All non-key attributes are fully functionally dependent on the entire primary key
   Goal: Remove partial dependencies (attributes depending on only part of a composite primary key)

1NF: First Normal Form

A table is in 1NF if:

1. All columns contain atomic (indivisible) values
2. Each row is unique and can be identified with a primary key
3. There are no repeating groups
   goal: ensure all data is atomic and eliminate duplicate/repeating groups

Decomposition of a Relation Schema

Definition

- Suppose that relation R contains attributes A1,...,An. A decomposition of R consists of replacing R by two ore more relations such that
  - Each new relation scheme contains a subset of the attributes R
  - Every attributes of R appears as an attribute of one of the new relations
- Intuitively decomposing R means we will store instances of the relation schemas produced by the decomposition, instead of instances of R

Properties

Lossless Join Decomposition

A decomposition if lossless join if the original table can be reconstructed by joining the decomposed tables without losing any data. Thus a decomposition of a relation R into R_1 and R_2 is lossless join if:

$$R_1 \bowtie R_2 = R$$

Conditions

1. Overlap condition: $R1 \cap R2$ must not be empty
2. The attributes in $R1 \cap R2$ must contain a key of at least one of the decomposed relations

Correctness

1. The algorithm produces a lossless-join decomposition
2. The resulting relations are BCNF

Non-deterministic

1. **Multiple possible paths**: the path taken is not fixed in advance
2. **Unpredictable Results**: the output may vary across different runs, even with identical inputs

Algorithm: Lossless Join Decomposition

1. For every $\mathrm{FD}\ X \to Y$ that is defined on $R(Z)$ and violates BCNF, decompose $R(Z)$ into $R1(X+ \cap Z)$ and $R2((Z - X+) \cup X)$
2. Repeat step 1 until there is no violation

Breaking down on how to find R1:

1. Calculate the closure of X, X+
2. Then intersect X+ with original set Z, which will give you R1

Breaking down on how to find R2:

1. Again, calculate the closure of X, X+
2. Subtract X+ from the original set Z, and then union it with X
3. This will give you R2

Dependency Preserving Decomposition

A decomposition is dependency preserving if all functional dependencies in the original relation are preserved in the decomposed tables. Thus, given a relation R and a set of functional dependencies F, a decomposition into $R_1, R_2, \ldots, R_n$ is dependency preserving if:

$$(F_1 \cup F_2 \cup \ldots \cup F_n) \equiv F$$

It is not always possible to find a dependency preserving BCNF decomposition

Minimal Cover (canonical cover)

Question: Given a choice between dependencies that can be preserved, which ones do we really want to preserve?
Moral: Use the minimal set of FDs (minimal cover or canonical cover)

Algorithm: Finding the minimal cover of F

1. Convert FD's so that they have only one attribute on the right side
2. Remove all redundant attributes from the left sides
   - Remove an attribute if *closure on the left side* without the attribute includes the the attribute
3. Remove all redundant FDs

Breaking down step 2 (when you have multiple attributes on the left)

Suppose you have the FD $AB \rightarrow C$.
Let us check A

1. Check if $A \rightarrow C$ holds (i.e., C is in the closure of A, $C \in A+$)
2. If so, then we can drop B in the FD $AB \rightarrow C$. Thus, we just get $A \rightarrow C$, no need to check B.
3. Otherwise, Check B
   Let us check B
4. Check if $B \rightarrow C$ holds (i.e., C is in the closure of B, $C \in B+$)
5. If so, then we can drop A in the FD $AB \rightarrow C$. Thus, we just get $B \rightarrow C$, no need to check A.
6. Otherwise, Check A
   If neither checks are satisfied, then keep the FD $AB \rightarrow C$!

Lossless join and Dependency Preserving Decomposition into 3NF

Algorithm

1. Given a relation R with a minimal set of FDs F
2. Find a lossless-join BCNF decomposition of R
3. For every FD $X \rightarrow A$ in F, which is not preserved after the decomposition, create a relation with schema XA
4. Of the two relation schemes R1(X) and R2(Y) where $X \subset Y$, remove relation schema R1(X)