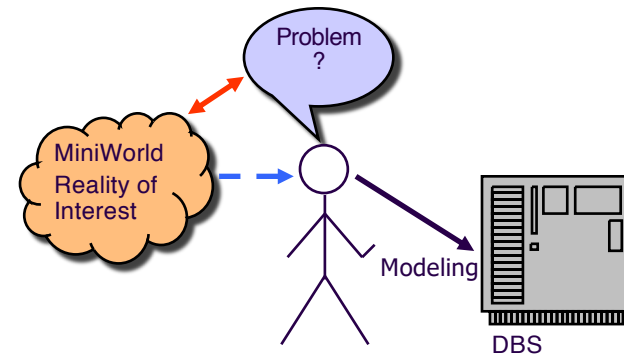
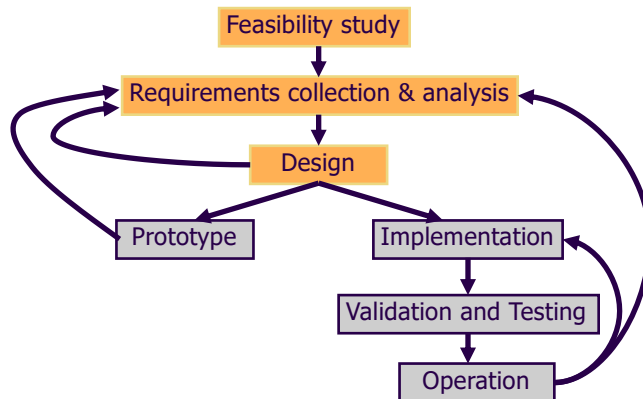


Database System Design

System / Data Modeling



Database System Life Cycle



Design

- Database Design
 - Database design is the activity of specifying the schema of a database in a given data model
- Functional Design

Functional Design

- High-level specification of Transactions
 - DBMS-independent
 - Even diagrams, UML
- Application program design
 - DBMS-specific (db Schema together with DML)
 - Language and environment-specific

Relational Database Design: The Good, The Bad and The Ugly

- ◆ Bad design & anomalies
 - ◆ Normal forms
- ◆ Universal Relational Approach
- ◆ Decomposition – normalization

Relational Database Design

- One single, large table
- Simple ?
- Good ? or Bad? Or just Ugly?



Goal

- Design 'good' tables
 - define what 'good' means
 - fix 'bad' tables
- in short: "we want tables where the attributes depend on the primary key, on the whole key, and nothing but the key"
- Let's see why, and how:

Bad Design

- ❑ Relation Schema:
takes1 (ssn, cid, grade, name, address)
- ❑ 'Bad' - why?
- ❑ because: ssn → (name, address)

ssn	cid	grade	name	address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

Pitfalls

- ❑ Redundancy
 - Waste of space (Repeated values, NULL)
 - Update anomalies (inconsistencies)

ssn	cid	grade	name	address
123	413	A	smith	Elm
123	415	B	smith	Main
123	211	A	smith	Main

❖ Insertion & Deletion anomalies

Insertion Anomaly

- ❑ "jones" registers, but takes no class
- ❑ Where do you store his address!?!?

ssn	cid	grade	name	address
123	413	A	smith	Main
...
234	null	null	jones	Forbes

Deletion Anomaly

- ❑ delete the last class record of 'smith'
- ❑ What about the address !?!
 - (we lose his address!)

ssn	cid	grade	name	address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

Solution: decomposition

- Split offending table in two (or more)

ssn	cid	grade	name	address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main



Functional dependencies

- How do we split a bad table?
- How do we recognize a bad table?
- Set of rules: *normal forms*
- Functional Dependencies (FD)**
 - Let $R=(A_1, A_2, \dots, A_n)$ and $X \subseteq R$ and $Y \subseteq R$
 $X \rightarrow Y$ if the value of X **uniquely** determines a value of Y
 - X functionally determines Y
 - Y is functionally dependent on X

Functional Dependencies Examples

- They are not a property of a particular relation state
 - Property of R , not $r(R)$
- They cannot be automatically inferred from $r(R)$
 - It can show which dependencies may exist
 - And show which dependencies cannot exist
 - Must have knowledge of the semantics of the attributes of R for the full story

Functional Dependency Examples

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d2
a2	b2	c2	d2
a2	b2	c2	d3
a3	b3	c2	d4

- $A \rightarrow C$? **YES**
- $C \rightarrow A$? **NO (line 5)**
- $B \rightarrow C$? **NO (line 3)**
- $D \rightarrow B$? **YES**
- $AB \rightarrow D$? **NO (line 4)**

- Loans (loan_number, branch, customer, amount)
 - $\text{loan_number} \rightarrow \text{amount}$ is satisfied
 - $\text{loan_number} \rightarrow \text{branch}$ is satisfied
 - $\text{amount} \rightarrow \text{branch}$ is not satisfied

Decompositions

- There are careless, 'bad' decompositions
- There are correct decompositions
 - lossless
 - dependency preserving



Decomposition: lossless

- Non-lossy decomposition is called **lossless** or **non-additive** decompositions
- Definition:
 - Let R schema R, with FD 'F'.
 - R1, R2 is a lossless decomposition of R if for all relations r(R), r1(R1) and r2(R2)
$$r1 \bowtie r2 = r$$

Decomposition: lossy

R1(ssn, grade, name, address)

ssn	grade	name	address
123	A	smith	Main
123	B	smith	Main
234	A	jones	Forbes

R2(cid, grade)

cid	grade
413	A
415	B
211	A

ssn	cid	grade	name	address
123	413	A	smith	Main
123	415	B	smith	Main
234	211	A	jones	Forbes
123	211	A	Smith	Main

ssn → (name, address)

(ssn, cid) → grade ❌

can not recover original table with a join!

Decomposition: lossless...

- What about an easier criterion?
- Theorem:
 - A decomposition is lossless if the **joining attribute is a superkey** in at least one of the new tables
- Formally:
 - $R1 \cap R2 \rightarrow R1$ or
 - $R1 \cap R2 \rightarrow R2$

Example: Loseless Decomposition

R1

ssn	cid	grade
123	413	A
123	415	B
234	211	A

$(ssn, cid) \rightarrow grade$

R2

ssn	name	address
123	smith	Main
234	jones	Forbes

$ssn \rightarrow (name, address)$

ssn	cid	grade	name	address
123	413	A	smith	Main
123	415	B	smith	Main
234	211	A	jones	Forbes

$ssn \rightarrow (name, address)$

$(ssn, cid) \rightarrow grade$

Dependency Preservation & Canonical Cover

- ❑ We don't want the original FDs to span two tables.
- ❑ More specifically: ... the FDs of the **canonical cover**
- ❑ **Canonical Cover** is the **minimum set of FDs without any trivial, extraneous and implied FDs**
- ❑ Example:
 - fd.1 $A \rightarrow B$
 - fd.2 $B \rightarrow C$
 - fd.3 $AB \rightarrow C$
 - What is the canonical cover?
 - Keep only fd.1 & fd.2: $A \rightarrow B$ & $B \rightarrow C$

Rules of Inference: Armstrong's Axioms

- ❑ IR1 - **Reflexivity** rule:
 - if $B \subset A$, then $A \rightarrow B$
 - e.g., If $\{ssn, name\} \supseteq name$ then $\{ssn, name\} \rightarrow name$
- ❑ IR2 - **Augmentation** rule:
 - if $A \rightarrow B$, then $AC \rightarrow BC$
 - e.g., if $ssn \rightarrow name$ then $\{ssn, age\} \rightarrow \{name, age\}$
- ❑ IR3 - **Transitive** rule:
 - if $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$
 - e.g., if $ssn \rightarrow Dno$ and $Dno \rightarrow Dname$ then $ssn \rightarrow dname$

More Rules of Inference

- ❑ IR4 - **Decomposition** rule:
 - if $A \rightarrow BC$, then $A \rightarrow B$ and $A \rightarrow C$
 - e.g., if $ssn \rightarrow \{name, age\}$ then $ssn \rightarrow name$ and $ssn \rightarrow age$
- ❑ IR5 - **Union** rule:
 - if $A \rightarrow B$ and $A \rightarrow C$, then $A \rightarrow BC$
 - e.g., if $ssn \rightarrow name$ and $ssn \rightarrow age$ then $ssn \rightarrow \{name, age\}$
- ❑ IR6 - **Pseudotransitivity**, Composition:
 - if $A \rightarrow B$ and $CB \rightarrow D$ implies $CA \rightarrow D$
 - e.g., if $isbn \rightarrow title$ and $\{author, title\} \rightarrow pubdate$ then $\{author, isbn\} \rightarrow pubdate$
- ❑ IR7 - **Self-determination**: $A \rightarrow A$

Decomposition: non-dependency preserving

- Dependency preserving: the original FDs should not span across tables
- Non-dependency preserving example:

S#	address	status
123	London	E
125	Paris	E
234	Pgh	A

$S\# \rightarrow (\text{address}, \text{status})$
 $\text{address} \rightarrow \text{status}$

S#	address	S#	status
123	London	123	E
125	Paris	125	E
234	Pgh	234	A

$S\# \rightarrow \text{address}$ $S\# \rightarrow \text{status}$

quiz: is it lossless?

Dependency Preserving Decomposition

Example

S#	address	status
123	London	E
125	Paris	E
234	Pgh	A

$S\# \rightarrow (\text{address}, \text{status})$
 $\text{address} \rightarrow \text{status}$

S#	address	address	status
123	London	London	E
125	Paris	Paris	E
234	Pgh	Pgh	A

$S\# \rightarrow \text{address}$ $\text{address} \rightarrow \text{status}$
 but: $S\# \rightarrow \text{status} ?$

♦ why is dependency preservation good?

Why Dependency Preservation?

- Record the status for Athens to be A:

S#	address	status
123	London	E
125	Paris	E
234	Pgh	A

$S\# \rightarrow (\text{address}, \text{status})$
 $\text{address} \rightarrow \text{status}$

S#	address	address	status
123	London	London	E
125	Paris	Paris	E
234	Pgh	Pgh	A
333	Athens	Athens	A

$S\# \rightarrow \text{address}$ $\text{address} \rightarrow \text{status}$

Decomposition - conclusions

- decompositions should always be lossless
 - joining attribute \rightarrow superkey
- Decompositions should be dependency preserving whenever possible

Testing for Loseless Decomposition

□ Given

- relation schema $R(A_1, A_2, \dots, A_n)$,
- a set of functional dependencies F and
- decomposition $\rho = \{R_1, R_2, \dots, R_m\}$

□ Steps:

- Construct an $m \times n$ table S ,
with a column for each of the n attributes in R and
row for each of the m relations in the decomposition.

Testing for Loseless Decomposition...

- For each cell $S(i,j)$ of S ,
If the attribute for the column A_j is in the relation
for the row R_i ,
then set $S(i,j) = k(j)$ to indicate *known* value
else set $S(i,j) = u(i,j)$ to indicate *unknown* value
- Consider each FD, $X \rightarrow Y \in F$ until no more changes can
be made to S .
 - Look for rows whose X -column agrees
 - EQUATE Y -column
- After all possible changes have been made to S ,
 - If a row is made up entirely of symbols $k(1), k(2), \dots, k(n)$
the join is lossless.
 - Otherwise, if there is no such row, the join is lossy.

Example

- $R(A, B, C, D, E)$
- Decomposition: $R_1(A, C), R_2(A, B, D), R_3(D, E)$
- FDs: $A \rightarrow C, AB \rightarrow D, D \rightarrow E$

	A	B	C	D	E
R1(A,C)					
R2(A,B,D)					
R3(D,E)					

Example

- $R(A, B, C, D, E)$
- Decomposition: $R_1(A, C), R_2(A, B, D), R_3(D, E)$
- FDs: $A \rightarrow C, AB \rightarrow D, D \rightarrow E$

	A	B	C	D	E
R1(A,C)	K	U	K	U	U
R2(A,B,D)	K	K	U	K	U
R3(D,E)	U	U	U	K	K

Example

- R (A,B,C,D,E)
- Decomposition: R1(A,C), R2(A,B,D), R3(D,E)
- FDs: $A \rightarrow C$, $AB \rightarrow D$, $D \rightarrow E$

	A	B	C	D	E
R1(A,C)	K	U	K	U	U
R2(A,B,D)	K	K	K	K	K
R3(D,E)	U	U	U	K	K

Example

- R (A,B,C,D,E)
- Decomposition: R1(A,C), R2(A,B,D), R3(D,E)
- FDs: $A \rightarrow C$, $AB \rightarrow D$, $D \rightarrow E$

	A	B	C	D	E
R1(A,C)	K	U	K	U	U
R2(A,B,D)	K	K	K	K	K
R3(D,E)	U	U	U	K	K