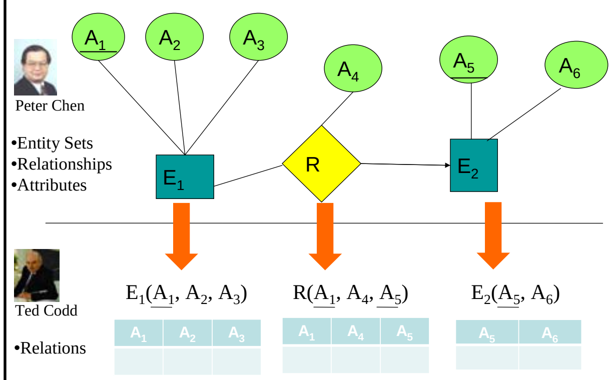
CS411
Database Systems

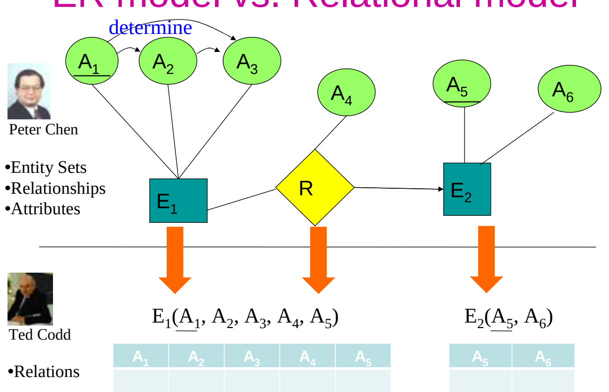
04: Relational Schema Design

Kazuhiro Minami

ER model vs. Relational model



ER model vs. Relational model



Reminder: redundancy causes trouble

SSN	Name	Phone Number
123-32-1099	Fred	(201) 555-1234
123-32-1099	Fred	(206) 572-4312
909-43-4444	Joe	(908) 464-0028
909-43-4444	Joe	(212) 555-4000
234-56-7890	Jocelyn	(212) 555-4000

Potential Inconsistency

update anomaly = update one copy of Fred's SSN but not the other

deletion anomaly = delete all Fred's phones, lose his SSN as a side effect

Non-solution: multiple values in one field

<u>SSN</u>	Name	Phone Number
123-32-1099	Fred	(201) 555-1234 (206) 572-4312
909-43-4444	Joe	(908) 464-0028 (212) 555-4000

First Normal Form:
Only one value in each field.

Your common sense will tell you how to fix this schema

<u>SSN</u>	Name
123-32-1099	Fred
909-43-4444	Joe

<u>SSN</u>	<u>Phone Number</u>
123-32-1099	(201) 555-1234
123-32-1099	(206) 572-4312
909-43-4444	(908) 464-0028
909-43-4444	(212) 555-4000

No more update or delete anomalies.

What if you don't have common sense?

There is a theory to tell you what to do!

Original Schema **R**

normalize R

Transformed Schema **R***

R* must:

- Preserve the information of R
- Have minimal redundancy
- "Preserve dependencies":
  easy to check R's constraints
- Give good query performance
  (but the theory won't help you there)

This theory formalizes the concept of redundancy

Functional Dependencies

Functional dependencies generalize the idea of a key

If two tuples agree on the attributes $A_1, \ldots, A_n$, then they must also agree on attributes $B_1, \ldots, B_n$.

equiv

$A_1, \ldots, A_n$ **functionally determine** $B_1, \ldots, B_n$.

equiv

$A_1, \ldots, A_n \rightarrow B_1, \ldots, B_n$

EmpID	Name	Phone	Office
E0045	Alice	9876	SC 2119
E1847	Bob	9876	SC 2119A
E1111	Carla	9876	SC 2119A
E9999	David	1234	DCL 1320

EmpID $\rightarrow$ Name, Phone, Office

Office $\rightarrow$ Phone   Phone $\not\rightarrow$ Office

Name $\rightarrow$ EmpID isn't likely to hold in all instances of this schema, though it holds in this instance

More generally, an instance can tell you many FDs that don't hold, but not all those that do.

Use your common sense to find the FDs in the world around you

Product: name $\rightarrow$ price, manufacturer
Person: ssn $\rightarrow$ name, age
Company: name $\rightarrow$ stock price, president
School: student, course, semester $\rightarrow$ grade

We can define keys in terms of FDs

Key of a relation R is a set of attributes that
1. functionally determines all attributes of R
2. none of its proper subsets have this property.

Superkey = set of attributes that contains a key.

Reasoning with FDs

1) Closure of a set of FDs
2) Closure of a set of attributes

The closure S^+ of a set S of FDs is the set of all FDs logically implied by S.

R = {A, B, C, G, H, I}

S = {$A \rightarrow B$, $A \rightarrow C$, $CG \rightarrow H$, $CG \rightarrow I$, $B \rightarrow H$}

Does $A \rightarrow H$ hold?

You can prove whether it does!

Compute the closure S^+ of S using ***Armstrong's Axioms***

1. Reflexivity

   $A_1 \ldots A_n \rightarrow$ every subset of $A_1 \ldots A_n$

2. Augmentation

   If $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$,

   then $A_1 \ldots A_n C_1 \ldots C_k \rightarrow B_1 \ldots B_m C_1 \ldots C_k$

3. Transitivity

   If $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$ and $B_1 \ldots B_m \rightarrow C_1 \ldots C_k$,

   then $A_1 \ldots A_n \rightarrow C_1 \ldots C_k$

How to compute S^+ using Armstrong's Axioms

$S^+ = S$;

loop {

For each f in S,

apply the reflexivity and augmentation rules and add the new FDs to S^+.

For each pair of FDs in S,

apply the transitivity rule and add the new FDs to S^+

} until S^+ does not change any more.

You can infer additional rules from Armstrong's Axioms

Union

If $X \to Y$ and $X \to Z$, then $X \to YZ$
(X, Y, Z are sets of attributes)

Splitting

$X \to YZ$, then $X \to Y$ and $X \to Z$

Combining

$X \to Y$ and $X \to Z$, then $X \to YZ$

Pseudo-transitivity

$X \to Y$ and $YZ \to U$, then $XZ \to U$

The closure of a set of attributes contains everything they functionally determine

Given a set S of dependencies,

the closure of a set of attributes $\{A_1 \ldots A_n\}$,

written $\{A_1 \ldots A_n\}^+$,

is { B such that any relation that satisfies S also satisfies $A_1 \ldots A_n \to B$ }

It is easy to compute the closure of a set of attributes

Start with $X = \{A_1 \ldots A_n\}$.

repeat until X doesn't change **do**:

if $B_1 \ldots B_m \to C$ is in S,
and $B_1 \ldots B_m$ are all in X,
and C is not in X
then add C to X.

$A\ B \to C$
$A\ D \to E$
$B \to D$
$A\ F \to B$

$\{A, B\}^+ = \{A, B, C, D, E\}$
$\{A, F\}^+ = \{A, F, B, D, C, E\}$

What is the attribute closure good for?

1. Test if X is a superkey
   - compute X^+, and check if X^+ contains all attrs of R
2. Check if $X \rightarrow Y$ holds
   - by checking if Y is contained in X^+
3. Another (not so clever) way to compute closure S^+ of FDs
   - for each subset of attributes X in relation R, compute X^+ with respect to S
   - for each subset of attributes Y in X^+, output the FD $X \rightarrow Y$

Reminder: intended goals of schema refinement

- Minimize redundancy
- Avoid information loss
- Easy to check dependencies
- Ensure good query performance

Normal Forms

First Normal Form = all attributes are atomic
Second Normal Form (2NF) = obsolete

Boyce Codd Normal Form (BCNF)
Third Normal Form (3NF)
Fourth Normal Form (4NF)

Others...

Boyce-Codd Normal Form

A relation R is in BCNF if whenever there is a nontrivial FD $A_1 \ldots A_n \rightarrow B$ for R, $\{A_1 \ldots A_n\}$ is a superkey for R.

An FD is *trivial* if all the attributes on its right-hand side are also on its left-hand side.

SSN	Name	Phone Number
123-32-1099	Fred	(201) 555-1234
123-32-1099	Fred	(206) 572-4312
909-43-4444	Joe	(908) 464-0028
909-43-4444	Joe	(212) 555-4000
234-56-7890	Jocelyn	(212) 555-4000

What are the nontrivial functional dependencies?

SSN → Name (plus the FDs that can be derived from that)

What are the keys?

The only key is {SSN, Phone Number}.

How do I know? Augmentation + minimality.

Is it in BCNF?

No. SSN is not a key.

What if we are in a situation where Phone Number → SSN?

What are the nontrivial FDs?

Phone Number → SSN

SSN → Name

(plus FDs derived from these)

What are the keys?

Only {Phone Number}.

How do I know?

Augmentation, transitivity, minimality.

Is it in BCNF?

No.

SSN	Name	Phone Number
123-32-1099	Fred	(201) 555-1234
123-32-1099	Fred	(206) 572-4312
909-43-4444	Joe	(908) 464-0028
909-43-4444	Joe	(212) 555-4000
234-56-7890	Jocelyn	(212) 555-9999

A relation R is in BCNF if whenever there is a nontrivial FD $A_1 \ldots A_n \rightarrow B$ for R, $\{A_1 \ldots A_n\}$ is a superkey for R.

What about that alternative schema we recommended earlier---are they in BCNF?

SSN	Name
123-32-1099	Fred
909-43-4444	Joe

SSN	Phone Number
123-32-1099	(201) 555-1234
123-32-1099	(206) 572-4312
909-43-4444	(908) 464-0028
909-43-4444	(212) 555-4000

For each relation:

What are its important FDs?

What are its keys?

Is it in BCNF?

A relation R is in BCNF if whenever there is a nontrivial FD A1 ... An → B for R, {A1 ... An} is a superkey for R.

What about that alternative schema we recommended earlier---are they in BCNF?

SSN	Name
123-32-1099	Fred
909-43-4444	Joe

SSN	Phone Number
123-32-1099	(201) 555-1234
123-32-1099	(206) 572-4312
909-43-4444	(908) 464-0028
909-43-4444	(212) 555-4000

Important FDS: SSN → Name

Keys: {SSN}.

Is it in BCNF? Yes.

If Phone Number → SSN holds

Important FDS:

Phone Number → SSN.

Keys: {Phone Number}

Is it in BCNF? Yes.

If Phone Number → SSN doesn't hold

Important FDS: none.

Keys: {SSN, Phone Number}

Is it in BCNF? Yes.

What about that alternative schema we recommended earlier---are they in BCNF?

SSN	Name
123-32-1099	Fred
909-43-4444	Joe

SSN	Phone Number
123-32-1099	(201) 555-1234
123-32-1099	(206) 572-4312
909-43-4444	(908) 464-0028
909-43-4444	(212) 555-4000

True or False:

Any 2-attribute relation is in BCNF.