CSE 344: Section 10
Design Theory

November 30th, 2017

Administrivia

Web quiz 6 due tomorrow!

Web quiz 7 due next Tuesday (last one!)

HW 8 due next Friday (last one!)

Boyce-Codd Normal Forms (BCNF)

Motivation of decomposition:

1. Less redundancy
2. Easy to update
3. Easy to delete

Boyce-Codd Normal Form: The closure of every set of attributes is either itself or all attributes

Normal Forms

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

v.s.

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

+

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121

Lossless Decomposition

Decompositions: Breaking down a table by columns

Lossless Decomposition: Joining the decomposed tables becomes the exact original table

- All original rows included in joined tables

<u>Goal:</u> Check if there can be extra tuples

Chase Algorithm

Check if decomposition is lossless (use the tableau method for doing this)

R(A, B, C, D) decomposed into S1(A, D), S2(A, C), S3(B, C, D)
FDs: {A -> B, B -> C, CD -> A}

Initialize

	A	B	C	D
S1	a	b1	c1	d
S2	a	b2	c	d2
S3	a3	b	c	d

Apply A -> B

	A	B	C	D
S1	a	b1	c1	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply B -> C

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply CD -> A

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a	b	c	d

Chase Algorithm (walkthrough)

R(A, B, C, D) decomposed into S1(A, D), S2(A, C), S3(B, C, D)
FDs: {A -> B, B -> C, CD -> A}

Initialize

	A	B	C	D
S1	a	b1	c1	d
S2	a	b2	c	d2
S3	a3	b	c	d

Apply A -> B

	A	B	C	D
S1	a	b1	c1	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply B -> C

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply CD -> A

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a	b	c	d

Chase Algorithm (walkthrough)

R(A, B, C, D) decomposed into S1(A, D), S2(A, C), S3(B, C, D)
FDs: {A -> B, B -> C, CD -> A}

S1 and S2 agree on A

Set same B for S1 and S2

Initialize

	A	B	C	D
S1	a	b1	c1	d
S2	a	b2	c	d2
S3	a3	b	c	d

Apply A -> B

	A	B	C	D
S1	a	b1	c1	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply B -> C

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply CD -> A

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a	b	c	d

Chase Algorithm (walkthrough)

R(A, B, C, D) decomposed into S1(A, D), S2(A, C), S3(B, C, D)
FDs: {A -> B, B -> C, CD -> A}

Initialize

	A	B	C	D
S1	a	b1	c1	d
S2	a	b2	c	d2
S3	a3	b	c	d

Apply A -> B

	A	B	C	D
S1	a	b1	c1	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply B -> C

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply CD -> A

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a	b	c	d

Chase Algorithm (walkthrough)

R(A, B, C, D) decomposed into S1(A, D), S2(A, C), S3(B, C, D)
FDs: {A -> B, B -> C, CD -> A}

Initialize

	A	B	C	D
S1	a	b1	c1	d
S2	a	b2	c	d2
S3	a3	b	c	d

S1 and S2 agree on B

Apply A -> B

	A	B	C	D
S1	a	b1	c1	d
S2	a	b1	c	d2
S3	a3	b	c	d

Set same C for S1 and S2

Apply B -> C

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply CD -> A

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a	b	c	d

Chase Algorithm (walkthrough)

R(A, B, C, D) decomposed into S1(A, D), S2(A, C), S3(B, C, D)
FDs: {A -> B, B -> C, CD -> A}

Initialize

	A	B	C	D
S1	a	b1	c1	d
S2	a	b2	c	d2
S3	a3	b	c	d

Apply A -> B

	A	B	C	D
S1	a	b1	c1	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply B -> C

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a3	b	c	d

Apply CD -> A

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a	b	c	d

Chase Algorithm (walkthrough)

R(A, B, C, D) decomposed into S1(A, D), S2(A, C), S3(B, C, D)
FDs: {A -> B, B -> C, CD -> A}

Initialize

	A	B	C	D
S1	a	b1	c1	d
S2	a	b2	c	d2
S3	a3	b	c	d

Apply A -> B

	A	B	C	D
S1	a	b1	c1	d
S2	a	b1	c	d2
S3	a3	b	c	d

S1 and S3 agree on CD

Apply B -> C

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a3	b	c	d

Set same A for S1 and S3

Apply CD -> A

	A	B	C	D
S1	a	b1	c	d
S2	a	b1	c	d2
S3	a	b	c	d

Chase Algorithm

Why does this work? What's the intuition?

We know $R \subseteq S1 \bowtie S2 \bowtie \ldots$

Chase algorithm shows $R \supseteq S1 \bowtie S2 \bowtie \ldots$

$(R \subseteq S1 \bowtie S2 \bowtie \ldots) \land (R \supseteq S1 \bowtie S2 \bowtie \ldots) \rightarrow R \equiv S1 \bowtie S2 \bowtie \ldots$

Chase Algorithm

How does the chase algorithm shows R $\supseteq$ S1 $\bowtie$ S2 $\bowtie$ … ?

- We take some arbitrary tuple in S1 $\bowtie$ S2 $\bowtie$ … (our joined projections of R) and see if it is in R.
- Order of applying FDs does not matter because natural join is commutative.
- “Agreeing” when applying FDs is like saying “Since I’m natural joining and the attributes are definitely matching the dependencies must also match.”
- When a row matches the tuple in R this means our joining (i.e. recovery through knowing FDs) has resulted in a tuple in R