# Algorithms for decomposition

Introduction to Database Design 2011, Lecture 9



#### Overview

- Decomposition to BCNF
  - algorithm for lossless decomposition
- Decomposition to 3NF
  - algorithm for lossless and dependency preserving decomposition
- 4NF
- Course evaluation



# Mandatory assignments

 Final deadline for mandatory assignments 1-3 on April 12



# BCNF decomposition

- Compute F<sup>+</sup>
- Repeat the following while the schema is not BCNF
  - Find a BCNF violation  $A_1 A_2 ... A_n$  →  $B_1 B_2 ... B_m$  in schema R(α)
  - Decompose R into  $((\alpha-B_1\ B_2\ ...\ B_m)\cup A_1\ A_2\ ...A_n)$  and  $(A_1\ A_2\ ...A_n\ B_1\ B_2\ ...\ B_m)$



### Example

Decompose the relation

cd\_shop(cd\_id, artist, title, order\_id, order\_date, quantity, customer\_id, name, address)

With the functional dependencies

```
cd_id → artist, title
  customer_id → name, address
order_id → order_date, customer_id
  order_id, cd_id → quantity
```



# Non determinancy

- Much depends on the choice of BCNF violation
- Try e.g. decomposing first using

```
order_id → order_date, customer_id
```

- There is no guarantee that decomposition is dependency preserving
- (even if there is a dependency preserving decomposition)
- One heuristic is to maximise right hand sides of BCNF violations



#### Correctness

- Tables become smaller for every decomposition
- Every 2-attribute table is BCNF
- So in the end, the schema must be BCNF
- Every decomposition is lossless by rule mentioned
   2 weeks ago (book page 346)



# 3NF decomposition

- Compute a canonical cover
- Create a table  $(A_1 A_2 ... A_n B_1 B_2 ... B_n)$  for every dependency  $A_1 A_2 ... A_n \rightarrow B_1 B_2 ... B_n$  in cover
- If no table contains a candidate key
  - add a table whose attributes is a candidate key
- Optional: erase unnecessary tables



# Example

Decompose the relation

cd\_shop(cd\_id, artist, title, order\_id, order\_date, quantity, customer\_id, name, address)

With the functional dependencies

```
cd_id → artist, title
  customer_id → name, address
order_id → order_date, customer_id
  order_id, cd_id → quantity
```

(note that these are a canonical cover)



# Alternative BCNF decomposition

- Example suggests the following alternative algorithm for BCNF decomposition
  - Use 3NF decomposition
  - Do further BCNF decompositions if needed



# 3NF decomposition examples

dept\_advisor(s\_ID, i\_ID, dept\_name)

```
i_ID → dept_name
s_ID, dept_name → i_ID
```

 Variant: dept\_advisor(s\_ID, i\_ID, dept\_name, semester) (same dependencies)



# Example

- Employee of the month example for Big Kahuna Burger
- Table: (empl\_id, name, branch, year, month)
- Functional dependencies:

```
empl_id \rightarrow name, branch branch, year, month \rightarrow empl_id
```

Decompose to BCNF and to 3NF



#### Correctness

- Decomposition is lossless:
  - At least one schema contains candidate key
  - Losslessness follows from generalisation of "losslessness rule"
- Decomposition is dependency preserving
  - Each dependency in cover can be checked on one relation
- For proof of 3NF see book (slightly difficult)



# On using the decomposition algorithms

- Could use decomposition to design databases
- First find all necessary attributes and functional dependencies
- Decompose to 3NF or BCNF
- I do not recommend this!
- Much better to think in terms of entities and relations
- But algorithms are good to know if you encounter redundancy problems



# 4NF



# Example

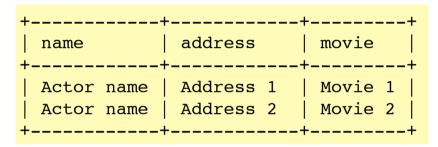
 Consider a database storing information about movie stars

- Clearly lots of redundancy here
- But no non-trivial functional dependencies!
- So BCNF

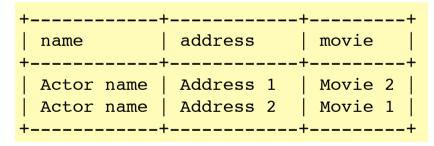


#### Problem in a nutshell

- Attributes address and movie are independent and not determined by other attributes
- For every pair of tuples



There are also tuples



This is called a multivalued dependency



# Multivalued dependencies

- Consider a table  $R(\alpha\beta\gamma)$
- **Definition.** There is a multivalued dependency  $\alpha \rightarrow \beta$  if for all tuples t,u in all legal instances
  - if  $t[\alpha] = u[\alpha]$
  - then there exists tuple s such that

$$s[\alpha] = t[\alpha]$$
  

$$s[\beta] = t[\beta]$$
  

$$s[\gamma] = u[\gamma]$$

In example name → address



# Rules for multivalued dependencies

- In  $R(\alpha\beta\gamma)$  if  $\alpha \rightarrow \beta$  then also  $\alpha \rightarrow \gamma$
- If  $\alpha \rightarrow \beta$  then also  $\alpha \rightarrow \beta$  (can take s = u)
- Consequences
  - if  $\beta \subseteq \alpha$  then  $\alpha \rightarrow \beta$
  - = if  $\alpha$  superkey then  $\alpha \rightarrow \beta$
- Transitivity: if  $\alpha \rightarrow \beta$  and  $\beta \rightarrow \gamma$  then  $\alpha \rightarrow \gamma$
- It is **not** the case that if  $\alpha \rightarrow \beta \gamma$  then  $\alpha \rightarrow \beta$



#### 4NF

- **Definition.** A table r(R) is in **4NF** if for all multivalued dependencies  $\alpha \rightarrow \beta$  either
  - $\beta \subseteq \alpha$  ( $\alpha \rightarrow \beta$  is trivial)
  - or α is a superkey
- Definition. A schema is in 4NF if all tables are in 4NF
- Theorem. A schema in 4NF is also BCNF



# 4NF decomposition

- There is a lossless decomposition algorithm for 4NF
- It is the same as the one for BCNF but uses multivalued dependencies



#### Normal forms

- A tower of normal forms
  - 4NF
  - BCNF
  - **-** 3NF
  - **-** 2NF
  - INF
- Any schema satisfying a normal form also satisfies the ones below
- (there do exist even higher normal forms)



# Summary

- Algorithm for lossless decomposition into BCNF
- Algorithm for lossless and dependency preserving decomposition into 3NF
- Even BCNF schemes may have redundancy
- 4NF normalisation gets rid of even more redundancy

