برای جلسه آنلاین فصل هفت

یکشنبه ۱۴ آذر ۱۴۰۰

مرور کلی اصول و مهارتهای لازم در نرمال سازی

۱) مبانی نرمال‌سازی

a. **ورودیهای ما**

i. **یک یا چند** schema **غیربهینه (شاید حاصل یک طراحی قدیمی)**

ii. **یک مجموعه قواعد در قالب** FDها **(برگرفته از سازمان)**

۱. **نوعی تعمیم مفهوم کلید، و قدرت کلید در یک** schema **در تعیین بقیه** attributeها

b. **خروجی مورد انتظار**

i. **یک یا چند** schema **که در سطوح نرمال (بهینه) باشند**

۱. **در عمل یکی از دو سطح** BCNF **یا** 3NF

۲) تکنیکهایی که لازم است بلد باشید

a. **محاسبه بستار (Closure) یک مجموعه** FDها

i. **ورودی:** F **خروجی:** F^+

ii. **استفاده از سه قاعده آرمسترانگ:**

- **Reflexive rule:** if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
- **Augmentation rule:** if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$
- **Transitivity rule:** if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

iii. **و سه قاعده مکمل:**

- **Union rule**: If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds.
- **Decomposition rule**: If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds.
- **Pseudo-transitivity rule**: If $\alpha \rightarrow \beta$ holds and $\gamma\,\beta \rightarrow \delta$ holds, then $\alpha\,\gamma \rightarrow \delta$ holds.

iv. **الگوریتم محاسبه بستار** FD**:**

```
F+ = F
repeat
    for each functional dependency f in F+
        apply reflexivity and augmentation rules on f
        add the resulting functional dependencies to F+
    for each pair of functional dependencies f1 and f2 in F+
        if f1 and f2 can be combined using transitivity
            then add the resulting functional dependency to F+
until F+ does not change any further
```

b.

c. **بستار** attribute

i. **ورودی:** α **خروجی:** α^+

ii. **الگوریتم محاسبه بستار صفت آلفا:**

```
result := α;
while (changes to result) do
    for each β → γ in F do
        begin
            if β ⊆ result then result := result ∪ γ
        end
```

iii. یکی از کاربردها: مشخص کردن این که آلفا کلید است یا نه؟ میتواند همه را تعیین کند؟

d. حذف attributeهای اضافی از FDها

i. از سمت راست

- For example, if we have AB → CD and remove C, we get the possibly weaker result AB → D. It may be weaker because using just AB → D, we can no longer infer AB → C.

ii. از سمت چپ

- For example, if we have AB → C and remove B, we get the possibly stronger result A → C. It may be stronger because A → C logically implies AB → C, but AB → C does not, on its own, logically imply A → C

iii. الگوریتم چک کردن اضافی بودن attribute

- To test if attribute $A \in \beta$ is extraneous in β
  - Consider the set:
    $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$,
  - check that α^+ contains A; if it does, A is extraneous in β
- To test if attribute $A \in \alpha$ is extraneous in α
  - Let $\gamma = \alpha - \{A\}$. Check if $\gamma \rightarrow \beta$ can be inferred from F.
    - Compute γ^+ using the dependencies in F
    - If γ^+ includes all attributes in β then , A is extraneous in α

e. محاسبه پوش کانونی یک مجموعه از FDها

i. ورودی: F خروجی: F_c

ii. حذف attributeهای اضافی

iii. یکتاسازی سمت چپ قواعد

iv. الگوریتم پوش کانونی:

repeat

Use the union rule to replace any dependencies in F of the form

$\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$

Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous attribute either in α or in β

/* Note: test for extraneous attributes done using F_c not F*/

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$

until (F_c not change

f. چک کردن تجزیه بدون گم‌شدگی – Lossless decomposition (شرط الزامی در تجزیه)

i. لازم است که:

- $R_1 \cap R_2 \rightarrow R_1$
- $R_1 \cap R_2 \rightarrow R_2$

g. چک کردن حفظ وابستگی – dependency preserving (شرط مطلوب در تجزیه)

Let F_i be the set of dependencies F^+ that include only attributes in R_i.

- A decomposition is **dependency preserving**, if

  $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$

h. تجزیه BCNF

i. حذف وابستگی‌های درونی هر schema تا برسیم به جایی که برای هر FD داشته باشیم:

- $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
- α is a superkey for R

ii. برای هر وابستگی درونی که توسط یک FD ایجاد شده، تجزیه به دو شمای زیر:

- $(\alpha \cup \beta)$
- $(R - (\beta - \alpha))$

iii. گامهای دقیق تجزیه BCNF:

result := {*R* };
done := false;
compute F^+;
while (not *done)* **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
        holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
          and $\alpha \cap \beta = \emptyset$;
        result := (*result* – R_i) $\cup$ (R_i – β) $\cup$ (α, β);
      end
    else *done* := **true;**

i. تجزیه 3NF

i. حذف وابستگی‌های درونی هر schema تا برسیم به جایی که برای هر FD داشته باشیم:

- $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \in \alpha$)
- α is a superkey for R
- Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

  (**NOTE**: each attribute may be in a different candidate key)

ii. گامهای دقیق تجزیه 3NF:

Let F_c be a canonical cover for F;
$i := 0$;
for each functional dependency $\alpha \rightarrow \beta$ in F_c **do**
  if none of the schemas R_j, $1 \le j \le i$ contains $\alpha \beta$
    then begin
      $i := i + 1$;
      $R_i := \alpha \beta$
    end
if none of the schemas R_j, $1 \le j \le i$ contains a candidate key for R
  then begin
    $i := i + 1$;
    R_i := any candidate key for R;
  end
/* Optionally, remove redundant relations */
repeat
if any schema R_j is contained in another schema R_k
  then /* delete R_j */
    $R_j = R_i$;
    $i=i-1$;
return $(R_1, R_2, ..., R_i)$

Back to the example:

f_1: *i_ID* → *dept_name*
f_2: *s_ID, dept_name* → *i_ID*

There are no extraneous attributes in any of the functional dependencies in F, so F_c contains f_1 and f_2. The algorithm then generates as R_1 the schema, (*i_ID dept_name*), and as R_2 the schema (*s_ID, dept_name, i_ID*). The algorithm then finds that R_2 contains a candidate key, so no further relation schema is created.

=

=