Cheat Sheet

Datamodelling and databases (Technische Universiteit Eindhoven)

FD's and MVD's

1. $\alpha \subseteq R \wedge \beta \subseteq R$, then $\alpha \to \beta$ {Reflexivity}
2. $\alpha \to \beta \wedge \gamma \subseteq R$, then $\gamma\alpha \to \beta\gamma$ {Augmentation}
3. $\alpha \to \beta \wedge \beta \to \gamma$, then $\alpha \to \gamma$ {Transitivity}
   a. $\alpha \to \beta \wedge \alpha \to \gamma$, then $\alpha \to \gamma\beta$ {union}
   b. $\alpha \to \beta\gamma$, then $\alpha \to \beta \wedge \alpha \to \gamma$
   c. $\alpha \to \beta \wedge \gamma\beta \to \delta$, then $\alpha\gamma \to \delta$

4. $\alpha \to\to \beta$, then $\alpha \to\to (R-\beta)-\alpha$ {Complementation}
5. $\alpha \to\to \beta \wedge \gamma \subseteq R \wedge \delta \subseteq \gamma$, then $\gamma\alpha \to\to \delta\beta$ {MVD Augmentation}
6. $\alpha \to\to \beta \wedge \beta \to\to \gamma$, then $\alpha \to\to \gamma - \beta$ {MVD Transitivity}
7. $\alpha \to \beta$, then $\alpha \to\to \beta$ {Replication}
8. $\alpha \to\to \beta \wedge \gamma \subseteq \beta \wedge \exists\delta: \delta \subseteq R \wedge \delta \cap \beta = \emptyset \wedge \delta \to \gamma$, then $\alpha \to \gamma$ {Coalescence}
   a. $\alpha \to\to \beta \wedge \alpha \to\to \gamma$, then $\alpha \to\to \beta \cup \gamma$ {MVD Union}
   b. $\alpha \to\to \beta \wedge \alpha \to\to \gamma$, then $\alpha \to\to \beta \cap \gamma$ {Intersection}
   c. $\alpha \to\to \beta \wedge \alpha \to\to \gamma$, then $\alpha \to\to \beta - \gamma \wedge \alpha \to\to \gamma - \beta$ {Difference}

Armstrong Relations

Regular

For every set of relations in F⁺, where the relation is not irrelevant (NO SUPERKEYS), the first column of the Armstrong relation is just all ones, and the second is only one for every relation set in the closure.

Strong

The Cartesian product of every set of tuples in the regular Armstrong relation.

Usage

Check the soundness of functional dependencies, if for any A1=A2 and B1<>B2, then A→B does NOT hold.

Tuple Relational Calculus (TRC)

TRC has the form: $\{t|p(t)\}$

Example

List of all customers who bought something impulsively

$\{t|\exists p \in purchase(t[cID] = p[cID] \wedge \nexists\, s \in shoppinglist(\dots))\}$

Dependency Preserving

Decompositions R_1 and R_2 are dependency preserved if in both R, the functional dependencies can be verified.

Algorithm

Result := α
While {changes to result}
    For {each R_i in decompostion}
        $t = (result \cap R_i)^+ \cap R_i$
        $result = result \cup t$
    If {result contains all attributes in β}
        then $\alpha \to \beta$ is preserved

Relational Algebra (RA)

←: Result of the right is assigned to the left

∩: Intersection:

⋈: Both are modified by carthesian product and set equal to ALL common variables

÷: Mostly used with the phrase "for all". Only columns remain for which every column the left side corresponds to one other column of the right part. i.e. the remaining columns in left should have all variables of the right.

BCNF

For each $\alpha \to \beta$ in F⁺ at least one of the following holds:

1. $\alpha \to \beta$ is trivial i.e. $\beta \subseteq \alpha$
2. α is a superkey for R

result := {R};
compute F⁺ (to check FD's i.e. β in α^+, then $\alpha \to \beta$ holds)
while {there is a schema R_i in result not in BCNF}
begin
    let $\alpha \to \beta$ be a nontrivial FD that holds on R_i s.t. $\alpha \to R_i$ is not in F⁺ and $\alpha \cap \beta = \emptyset$;
    result := $(result - \{R_i\}) \cup (\{R_i - \beta\} \cup \{\alpha \cup \beta\})$;
    (i.e. R_i=(α, β) and R_{i+1} = $(R - \beta)$)
end

Entity Relation (ER)

Relation (Ruit)
Entity (Square)
Primary Key (Underline)

Every: ==
One or more (or none): ---
Exactly one: →
Exists only if another exists: weak entity

Overlapping specialization: Can be both (seperate arrow)
Disjoint specialization: Can be only one (two headed arrow)

Translation to Relational Model

Entity set: schema with same attributes
Weak entity set: also includes primary key of identifying set

Many2Many: also attributes primary keys of participating sets
Many2One: Adding primary key of ONE to MANY
One2One: Either one can add the primary key of the other

-.-: R gets primary keys of both sides (both primary in R), E's don't change
-.->: R gets primary keys of both sides (only – primary in R)
=.->: R gets removed, = side gets primary key of -> and r related to R

3NF (also in BCNF)

For each $\alpha \to \beta$ in F⁺ at least one of the following holds:

1. $\alpha \to \beta$ is trivial i.e. $\beta \subseteq \alpha$
2. α is a superkey for R
3. Each attribute B in $\beta - \alpha$ is contained in a candidate key for R

Let F_C be the canonical cover for F;
i:=0;
No schema in the beginning.
for each FD $\alpha \to \beta$ in F_C do
{
    if {none of the schemas R_j, $1 \leq j \leq i$ contains $\alpha\beta$}
    {
        i++;
        R_i:=$\alpha\beta$;
    }
    if {none of the schemas R_j, $1 \leq j \leq i$ contains a candidate key for R}
    {
        i++;
        R_i:=any candidate key for R;
    }
    if $\{R_j \subseteq R_k$ for $j \neq k\}$
        remove R_j;
}

4NF (also in BCNF)

For every MVD in D⁺ $\alpha \to\to \beta$ at least one of the following should hold

1. $\alpha \to\to \beta$ is trivial ($\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
2. α is a superkey for schema R

result := {R};
done := false;
compute D⁺ (same as before only now with MVD)
while not done
{
    if there is a schema not in 4NF
    {
        let $\alpha \to\to \beta$ be a nontrivial MVD
        $result := (result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$
    }
}

SQL

Set Operations

Union, Intersect, Except

With

The with clause creates temporary views: with max_balance(value) as `subquery'.

Select

SQL allows duplicates: use `distinct' to get rid of it.
In the select clause, attribures are selected which are desired.
`Select *' gives all attributes
`Select avg' gives the average value
`Select min/max' gives the min/max value
`Select sum' gives the sum
`Select count' gives the number of values

From

In the from, all relations are given and taken the Cartesian product (×).

Where

Specifies the conditions
(<> = unequal)
    Between: value between x and y
    Like (for strings): street like %Name%
    In: new subquery
    Some: i.e. bigger than some in subquery
    All: i.e. bigger than all in subquery
    Exists: evaluates true or false for a new subquery
    Unique: tests for duplicates in subquery

Order by

Desc(ending) or asc(ending) order can be used, for alphabetical order and such.

Group by

Gives a list for a list i.e. number of depositors for each branch: group by branch_name.

View

Create: create view 'name' as 'subquery'
Update: insert into 'name of view' values ('name1', 'name2',…).

Trigger

Create trigger 'name' after update on 'something'
Referring to new row as nrow
Referring to old row as orow //if necessary
For each row when 'some condition'
Begin
    Insert into 'somewhere in database'
    'Subquery'
End

Canonical Cover

For F = {AB→CD,…}
If A is extraneous: check if B⁺ includes DC in unmodified set.
If C is extreneous: check if AB⁺ includes C in modified set.
Union rule: if same at the left → merge right.

Datalog

Datalog consists of a set of rules that define views.
Natural Joint (⋈) is always used for same variable names.

Example

Give the list of account numbers and balances larger than $700 in Perryridge
P700(A,B):- account(A, 'Perryridge', B), B>700
?P700(A,B)

Note that the levels are highers up, lowest down i.e.
P700(A,B):- account(…),B>700
Account(A,B,C):-…

Not

If you use not, every aspect in the not should be present in the other sets.

Recursion

For everyone who works for John (direct or indirect)
Works_for(X,Y):-manager(X,Y)
Works_for(X,Y):-manager(X,Z),works_for(Z,Y)