Some other normal forms

- Fifth Normal Form (5NF) or called Project-Join Normal Form (PJNF).
- Domain-Key Normal Form (DKNF)
- For your reading pleasure. They will not be covered/examined.

Fifth Normal Form (Project-Join Normal Form) (5NF, PJNF)

(will not be covered/examined)

There exist relation that cannot be non-loss decomposed into two relations, but can be non-loss decomposed into three or more relations.

Example Let us consider the relation

STOCK(Agent, Company, Product)

We assume that:

1. Agents represent companies.
2. Companies make products.
3. Agents sell products
4. **If an agent sells a product and he represents the company making that product, then he sells that product for that company.**

Note: It is an all key relation. There is no FD or MVD in the relation.

Relation instances:

STOCK (Agent,	Company,	Product)
a_1	c_1	p_1
a_1	c_2	p_1
a_1	c_1	p_3
a_1	c_2	p_4
a_2	c_1	p_1
a_2	c_1	p_2
a_3	c_2	p_4

REP (Agent, Company)

Agent	Company
a_1	c_1
a_1	c_2
a_2	c_1
a_3	c_2

MAKE (Company, Product)

Company	Product
c_1	p_1
c_1	p_2
c_1	p_3
c_2	p_1
c_2	p_4

SELL (Agent, Product)

Agent	Product
a_1	p_1
a_1	p_3
a_1	p_4
a_2	p_1
a_2	p_2
a_3	p_4

Notes: (1) There is no FD or MVD in the relation STOCK

(2) The relation is in 4NF.

(3) There are redundant data in the relation.

(4) However, the relation can be non-loss decomposed into 3 relations, namely

REP (Agent, Company)
MAKE (Company, Product)
SELL (Agent, Product)

Q: How do you know this?

(5) REP $\bowtie$ MAKE $\bowtie$ SELL = STOCK

Defn: Let R be a relation and $R_1, \ldots, R_n$ be a decomposition of R. We say that R satisfies the **join dependency** $*\{R_1, R_2, \ldots, R_n\}$ iff

$$\bowtie_{i=1}^{n} R_i = R$$

$$(\text{ or } R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n = R$$

$$\text{or } R_1 * R_2 * \ldots * R_n = R \;)$$

Defn: A join dependency (JD) is **trivial** if one of the R_i is R itself.

Note: When $n = 2$, the join dependency of the form $*\{R_1, R_2\}$ is equivalent to a multivalued dependency.

Example. The relation STOCK(Agent, Company, product) satisfies the join dependency:

$*\{R_1$(Agent, Company), R_2(Agent, Product), R_3(Company, Product)$\}$

However, there is **no MVD** in the relation.

Defn: A relation R is in **fifth normal form** (**5NF**) or called **Project-Join normal form** (**PJNF**) iff every non-trivial join dependency in R is implied by the candidate keys of R.

i.e. whenever a non-trivial join dependency $*\{R_1, R_2, \ldots, R_n\}$ holds in R, implies **every** R_i (all the attributes of R_i) is a superkey for R.

Example: The relation STOCK(Agent, Company, Product) is not in 5NF.

Results: (1) A 5NF relation is in 4NF.

❖ (2) Any relation can be non-loss decomposed into an equivalent collect of 5NF relations, if covering criteria (of FDs) is not required.

Example: The relation Stock can be non-loss decomposed into 3 relations:

REP (Agent, Company)

SELL (Agent, Product)

MAKE (Company, Product)

All are in 5NF.

Domain-Key Normal Form (DKNF)

(will not be covered/examined)

Note that FDs, MVDs and JDs are some sorts of **integrity constraints**. There are other types of constraints:

(1) Domain constraint - which specifies the possible values of some attribute.

E.g. The only colors of cars are blue, white, red, grey.

E.g. The age of a person is between 0 and 150.

(2) Key constraint - which specifies keys of some relation.

Note: All key declarations are FDs but not reverse.

(3) General constraints - any other constraints which can be expressed by the first order logic.

E.g. If the first digit of a bank account is 9, then the balance of the account is greater than 2500.

Defn: Let D, K, G be the set of domain constraints, the set of key constraints, and the set of general constraints of a relation R.

R is said to be in **domain-key normal form** (DKNF) if

$$D \cup K \text{ logically implies } G.$$

i.e. all constraints can be expressed by only domain constraints and key constraints.

Example. Let Acct(acct#, balance) with acct# $\rightarrow$ balance and a general constraint:

" if the first digit of an account is 9, then the balance of the account is $\geq$ 2500."

- Relation Acct is not in DKNF.
- To create a DKNF design, we split the relation horizontally into 2 relations:

Regular_Acct (acct#, balance)

Key = {acct#}

Domain constraint: the first digit of acct# is not 9.

Special_Acct (acct#, balance)

Key = {acct#}

Domain constraints:

(1) t he first digit of acct# is 9, and.

(2) balance $\geq$ 2500.

Both relations are in DKNF. **Why?**

All constraints can now be enforced as domain constraints and key constraints.

Q: How to enforce them?

Note: We can rewrite the definitions of PJNF, 4NF, and BCNF in a manner which shows them to be special case of DKNF.

E.g. Let $R=(A_1, \ldots, A_n)$ be a relation.

Let dom(A_i) denote the domain of attribute A_i and let all these domains be infinite.

Then all domain constraints D are of the from

$A_i \subseteq dom(A_i)$.

Let the general constraints be a set G of FDs and MVDs .

Let K be the set of key constraints.

R is in 4NF iff it is in DKNF with respect to D, K, G.

(i.e. every FD and MVD is implied by the domain constraints and key constraints.)

Note: PJNF and BCNF can be rewritten similarly.

Q: How about 3NF?

Theorem

Let R be a relation in which dom(A) is infinite for each attribute A.

If R is in DKNF then it is in PJNF.

Thus if all domains are infinite, then

$$\text{DKNF} \Rightarrow \text{PJNF} \Rightarrow \text{4NF} \Rightarrow \text{BCNF} \Rightarrow \text{3NF}$$