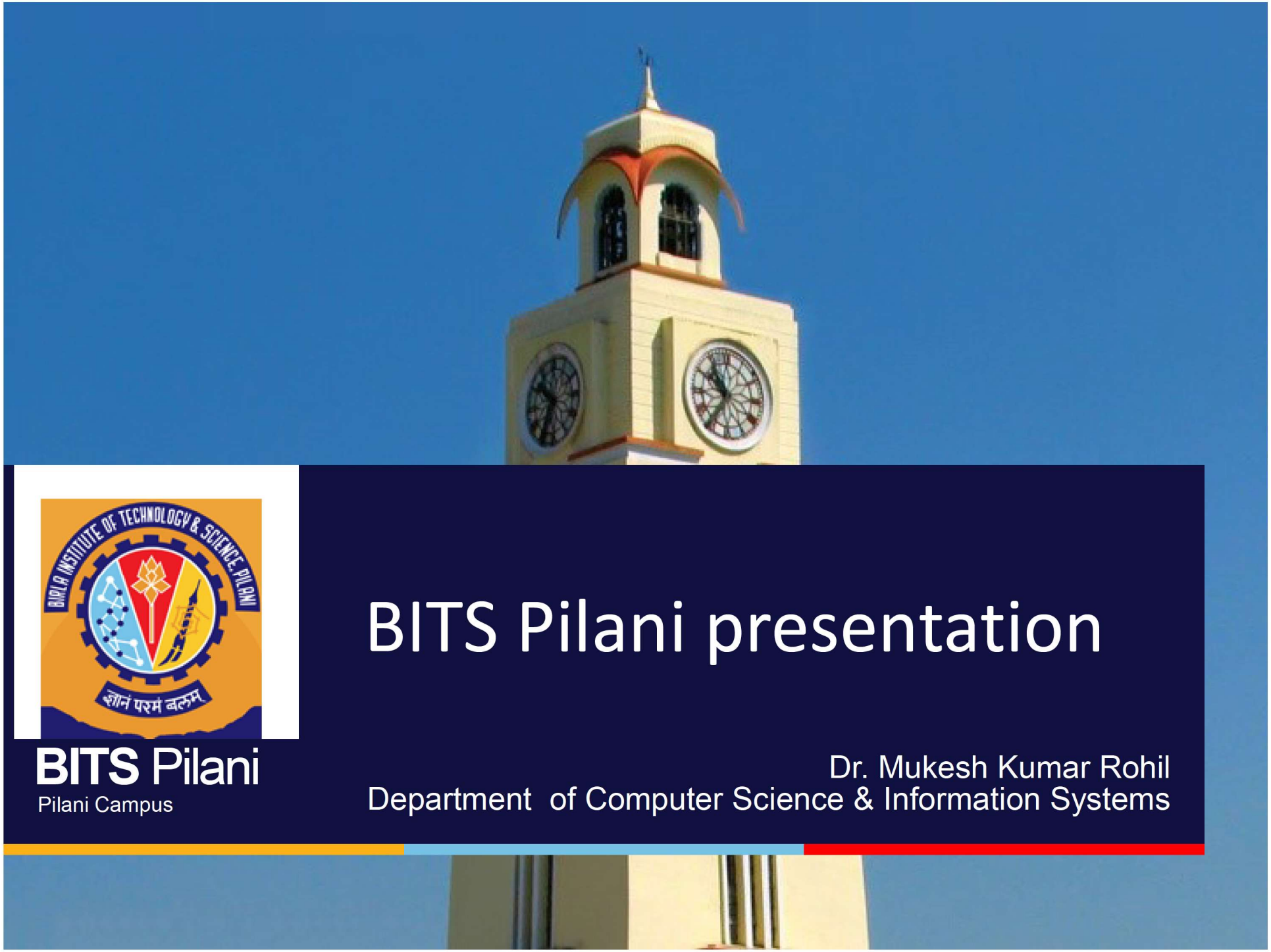

BITS Pilani presentation

Dr. Mukesh Kumar Rohil
Department of Computer Science & Information Systems

BITS Pilani
Pilani Campus

Introduction to Project-Join Normal Form (PJNF) and Domain-Key Normal Form (DKNF)

Database Systems: Introduction to PJNF and DKNF

Dr. Mukesh Kumar Rohil
Instructor of the course on
Database Systems

Computer Science &
Information Systems
Department

Birla Institute of
Technology & Science
Pilani – 333031
(Rajasthan)

Learning Objectives

- Fourth Normal Form is not the ultimate normal form?
- Normal Forms stronger than 4NF
- Project-Join Normal Form (PJNF or 5NF)
- Domain-Key Normal Form (DKNF)

The subsequent slides are adopted from:

1. https://www.comp.nus.edu.sg/~lingtw/onf.pdf
2. Reference Book R1:R1. Elmarsi R, & Navathe S B,***Fundamental of Database System***, 7e, Pearson Education, 2016.

Introduction

- The fourth normal form (4NF) is not the “ultimate” normal form.
- Multivalued dependencies help us understand and eliminate some forms of repetition (redundancy) of information that cannot be understood in terms of functional dependencies.
- There are types of constraints called join dependencies that generalize multivalued dependencies and lead to another normal form called **project-join normal form (PJNF)**. PJNF is also referred as 5NF in some literature.
- There is a class of even more general constraints that leads to a normal form called **domain-key normal form (DKNF)**.

Introduction …2

Some practical problems with the use of these generalized constraints are:

- These are hard to reason with
- There is no set of sound and complete inference rules for reasoning about these constraints.

=> Hence, PJNF and DKNF are used quite rarely.

Project-Join Normal Form (PJNF)

- There exist relations that cannot be non-loss decomposed into two relations, but can be non-loss decomposed into three or more relations.
- **Example Relation**: STOCK(Agent, Company, Product)
- **Four assumptions about the relation**:

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

PJNF …2 (Example Relation instances and loss-less join)

STOCK (Agent, Company, Product)

a1	c1	p1
a1	c2	p1
a1	c1	p3
a1	c2	p4
a2	c1	p1
a2	c1	p2
a3	c2	p4

Non-loss decomposition:

REP (Agent, Company)

a1	c1
a1	c2
a2	c1
a3	c2

MAKE (Company, Product)

c1	p1
c1	p2
c1	p3
c2	p1
c2	p4

SELL (Agent, Product)

a1	p1
a1	p3
a1	p4
a2	p1
a2	p2
a3	p4

PJNF …3

Observations from the previous slide:

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

(2) The relation, STOCK, is in 4NF.

(3) There are redundant data in the relation STOCK.

(4) The relation STOCK can be non-loss decomposed into 3 (three) relations, namely

REP(Agent, Company)

MAKE(Company, Product)

SELL(Agent, Product).

(5) REP ⋈ MAKE ⋈ SELL = STOCK

(where, ⋈ = Natural Join)

PJNF …4

Definition:

Let R be a relation and R_1 , …, R_n be a decomposition of R.

We say that R satisfies the join dependency *{ R_1 , R_2 , …, R_n }

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

(or $R_1 \bowtie R_2 \bowtie R_3 \ldots R_{n-1} \bowtie R_n = R$ or $R_1 * R_2 * \ldots * R_n = R$)

Definition:

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 *{R1 , R2 } is equivalent to a multivalued dependency.

PJNF …5

Example.

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

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

R_1 = REP, R_2 = MAKE, and R_3 = SELL.

Note: There is no MVD in the relation.

PJNF …6

Definition: A relation R is in fifth normal form (5NF) or 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, …, R_n} holds in R, implies every R_i (all the attributes of R_i) is a super-key for R.

Example:

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

Results considered without formal proof:

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

(2) Any relation can be non-loss decomposed into an equivalent collection 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.**

PJNF …7

Example:

SUPPLY(sname, partname, projname) // An all-key relation.

Additional Constraint to hold always:

Whenever a supplier s supplies part p, and a project j uses part p, and the supplier s supplies at least one part to project j, then supplier s will also be supplying part p to project j.

5NF: Three projections of SUPPLY relation as

R_1(sname, partname)

R_2(sname, projname)

R_3(partname, projname)

Domain-Key Normal Form (DKNF)

- FDs, MVDs and JDs are some sorts of integrity constraints.

- **There are other types of constraints:**

(1) Domain constraint (D) - 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 (K) - which specifies keys of some relation. All key declarations are FDs but not the reverse.

(3) General constraints (G) - 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.

Domain-Key Normal Form (DKNF) … 2

Definition:

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

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

D ∪ K logically implies G.

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

Domain-Key Normal Form (DKNF) … 3

Example:

Let Acct(acct#, balance) with acct# → balance and a general constraint: " if the first digit of an account is 9, then the balance of the account is ≥ 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) the first digit of acct# is 9, and

  (2) balance ≥ 2500.

Both relations are in DKNF.

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

Domain-Key Normal Form (DKNF) … 4

Note:

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

Example:

Let R=(A1 , …, An) be a relation.

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

Then all domain constraints D are of the from Ai ⊆ dom(Ai).

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.)

PJNF and BCNF can be rewritten similarly.

Domain-Key Normal Form (DKNF) … 5

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

DKNF $\Longrightarrow$ PJNF $\Longrightarrow$ 4NF $\Longrightarrow$ BCNF $\Longrightarrow$ 3NF

Thanks ….

Any questions please …..

Thanking you