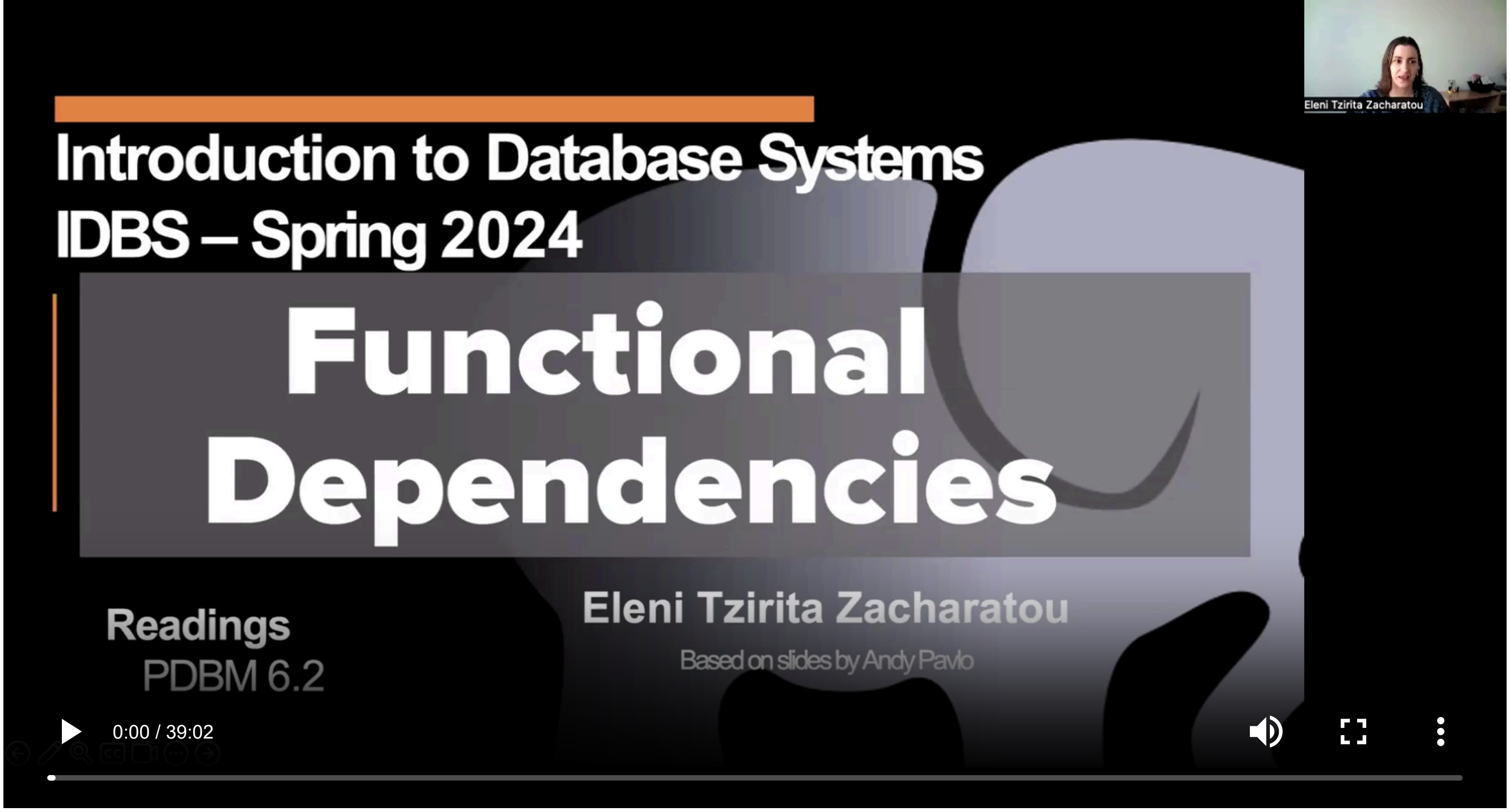
Normalization & Functional Dependencies



Redundancy Problems

Update Anomalies
→ If the room number changes, we need to make sure that we change all student records.
Insert Anomalies
→ May not be possible to add a student unless they are enrolled in a course.
Delete Anomalies
→ If all the students enrolled in a course are deleted, then we lose the room number.

Functional Dependencies

A Functional Dependency (FD) is a form of a constraint. It's part of a relation's schema to define a valid instance

Definition

$$X \to Y => (t_1[x] = t_{2[x]} => t_1[y] = t_{\prime\prime}[y])$$

If two tuples (t_1, t_2) agree on the X attribute, then they must agree on the Y attribute too.
You can check if an instance violates an FD, but you **cannot** prove that an FD is part of the schema using an instance.

The attribute that is **not** present in the right hand side of a Functional Dependency, will be present in the candidate key.

Trivial Dependency

A functional dependency $X \to Y$ where Y is a subset of X. ($Y \subseteq X$)

Prime Attributes

A **prime attribute** is part of a candidate key.

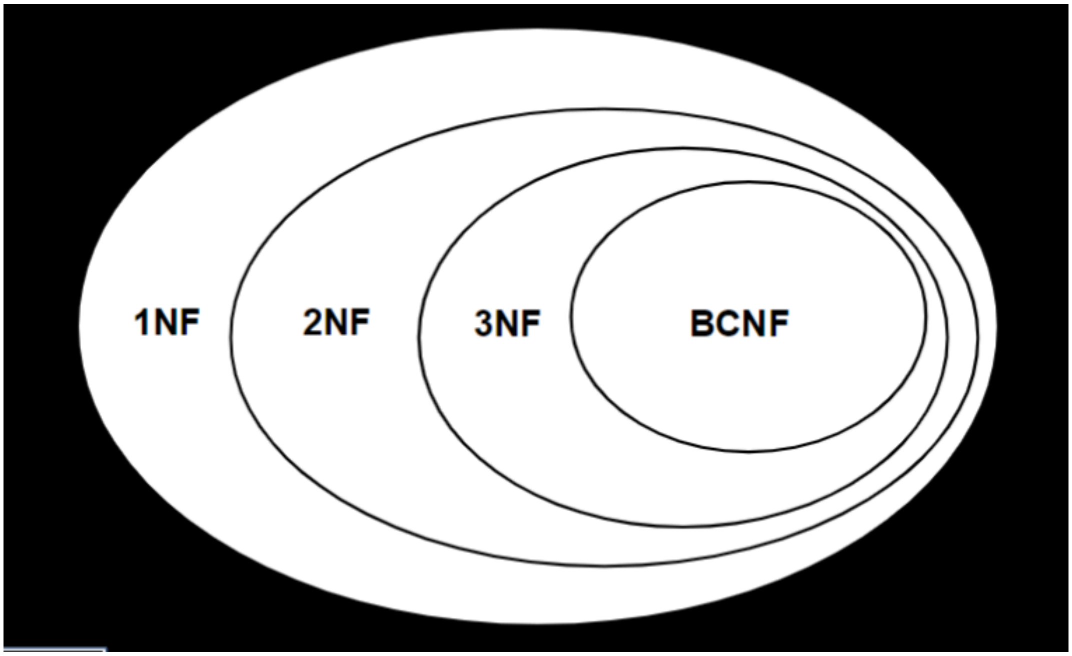
For example, in `R1(SSN, PNUMBER, PNAME, HOURS)`, both `SSN` and `PNUMBER` are prime, whereas `PNAME` and `HOURS` are not.

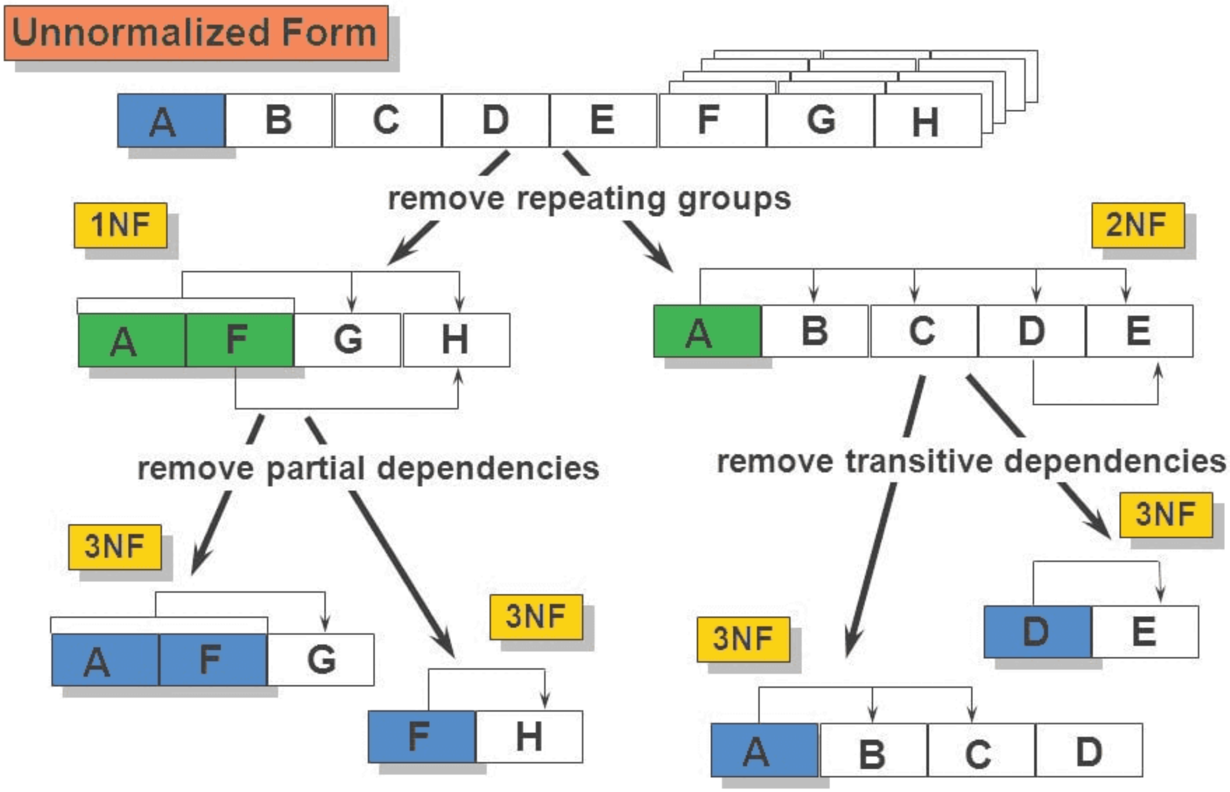
Normal forms

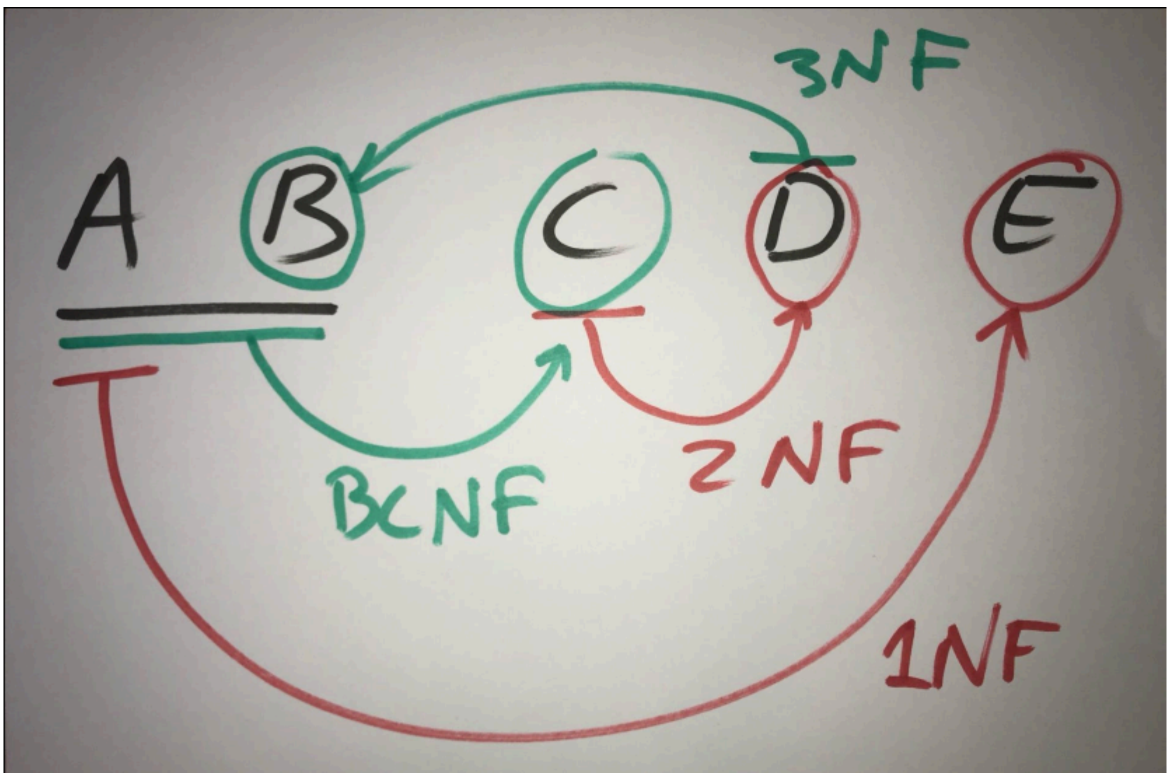
To identify "bad" Functional Dependencies - we check for **Normal forms** to ensure your relational schema design is free of certain issues functional dependencies are used to define normal forms.

The normal forms get more restrictive for each iterration of the normal forms -that is that Third normal form (3NF) is more restrictive than First normal form (1NF)

Venn Diagram







First normal form (1NF)

A relation is in 1NF if all its attribute types are atomic and single-valued:

- All rows must be unique (no duplicate rows)
- Each cell must only contain a single value (not a list)
- Each value should be non divisible (can't be split down further)
  Each [attribute]() in a database [Relations]() has a primitive type (atomic values), and a unique name.

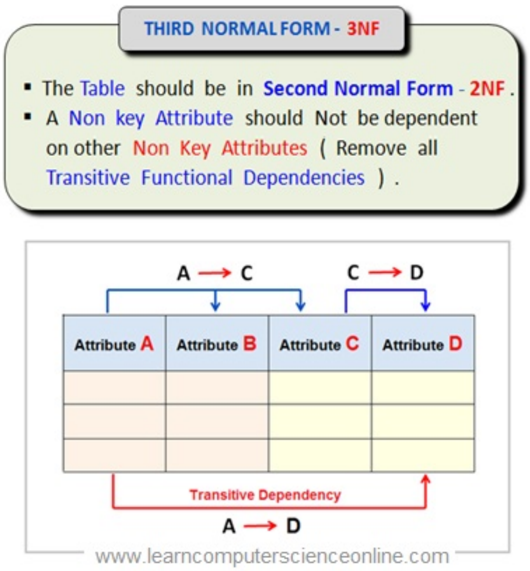
Second normal form (2NF)

First normal form (1NF) still applies +
non-key attributes **fully (not partially)** depend on the whole candidate key(s).

See Lecture 6, page 16

Third normal form (3NF)

A relation is in 3NF if it meets Second normal form (2NF) and has no Transitive dependencies on the primary key.



See also Lecture 6, page 18

Boyce-Codd normal form (BCNF)

BCNF is stricter than Third normal form (3NF), requiring each non-trivial dependency to have a Superkey on its left side.
A relational schema R is in Boyce–Codd normal form if and only if for every one of its dependencies $X \to Y$, at least one of the following conditions hold:[[

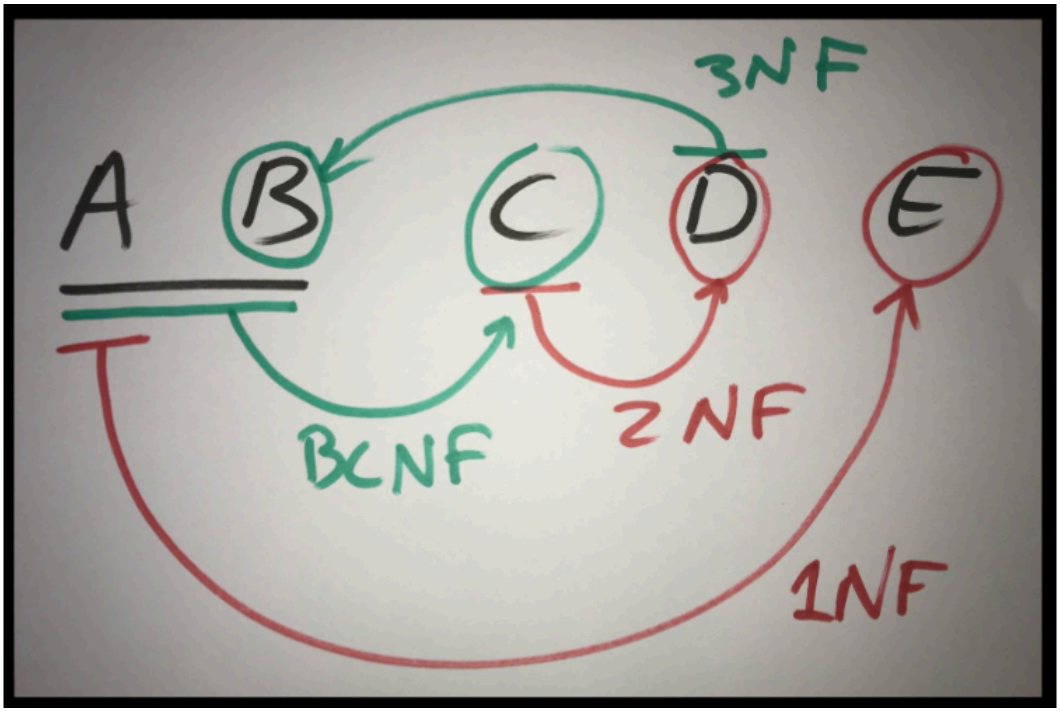
- $X \to Y$ is a Trivial Dependency,
- X is a superkey for schema R.

See Lecture 6, page 20

Normalization

1. Look for functional dependencies
2. Identify "bad" Functional Dependencies that break Normal forms

3. If there are such, then decompose the tables into two tables



Decomposition

1. Find all Functional Dependencies
2. While Functional Dependencies < Third normal form (3NF) exits $\rightarrow$ Decompose

Practical Decomposition

Consider relation R and (important) functional dependency $X \rightarrow Y$ that violates 3NF/BCNF
Decompose R into R_1 and R_2 where
$R_1 = R - Y$ (everything but Y = the right side)
$R_2 = \underline{X}Y$ (the whole FD = both left and right side)
This has the following nice properties
R_2 is (normally) in BCNF
Joining R_1 and R_2 (with = on all X attributes) yields R

Examples 1
Person($\underline{ID}$, $\mathrm{Name}, \mathrm{ZIP}, \mathrm{City}$),
$\mathrm{ZIP} \longrightarrow \mathrm{City}$
X = ZIP, Y = City
$R - Y$ = Person($\underline{ID}$, $\mathrm{Name}, \mathrm{ZIP}$)
XY = ZIP($\underline{ZIP}$, City)
Examples 2
$R = \underline{AB}CD$
$AB \rightarrow CD$
$C \rightarrow D$
Identify the Normal forms -> Second normal form (2NF)
New table: $\underline{C}D$
Update old table: $\underline{AB}C$
$AB \rightarrow D$ is stil valid trough transitivity of C: $AB \rightarrow C \rightarrow D$

See Lecture 6, page 26-32 for more examples