Databases 1

Daniel POP

Week 7/8

Database Design. Theory

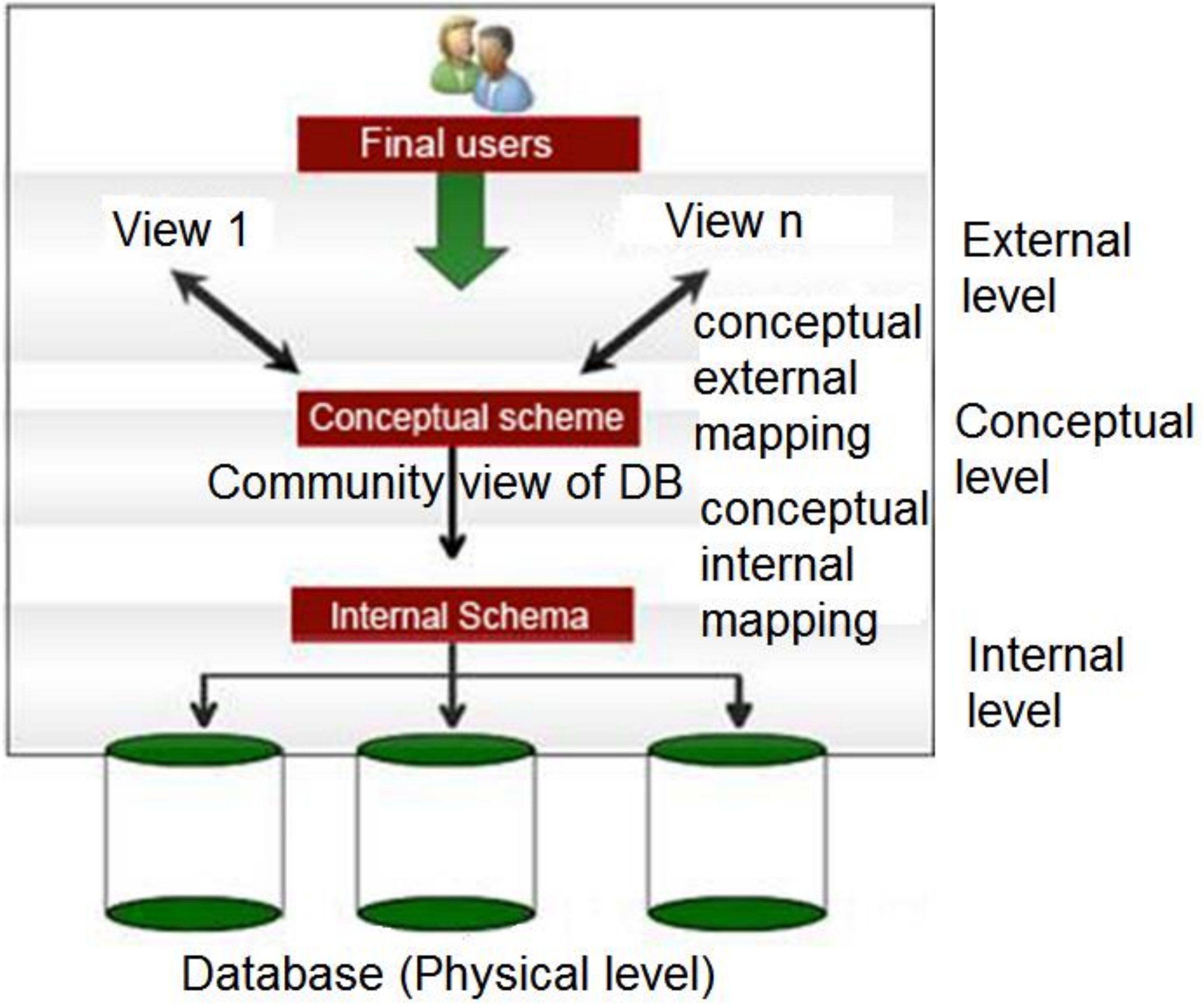
Agenda

- Database Design. Design Theory. An introduction
- Lossless join decomposition
- Functional dependencies. Attributes set closure
- Normal Forms. Normalization process
- 1NF. 2NF. 3NF
- BCNF
- Multi-valued dependencies
- 4NF
- 5NF
- De-normalization

Database design

- A major aim of a database system is to provide users with an abstract view of data, hiding certain details of how data is stored and manipulated.
- Can be complex
- Data first approach
- Poorly designed databases can have serious repercussions for the organization
- Top-down vs. bottom-up

ANSI/X3 SPARC Architecture for databases



The ANSI/X3 SPARC DBMS Framework: Report of the Study Group on Database Management Systems (1977)

Who are the users of the database

- Users are any application that accesses our database
  - The Mobile Application for Student Registration
  - The Web site used by teachers to enter students' grades
  - The video conferencing app that automatically stores attendance data in the database
  - The desktop application used by Accounting department to record details about students' grants
  - The DBA using SQL Server Management Studio / pgAdmin / Oracle Developer Suite / etc. to run database profiler or denormalize some tables
  - etc.

Reasons for separation

- Each user able to access the same data, but also able to have a personalized view of the data
- A user can change its view of the data without affecting other users
- Users should not have to deal directly with physical implementation details
- A DBA should be able to change the DB structure without affecting the users
- The internal structure of the database should not be affected by changes to the physical aspects of storage

External level (schema)

- **The user's view of the database. This level describes the part of the database that is relevant to each user**
- Users' requirements drive the model of the database; problem is users don't know what they need ☺
- Contains entities, attributes, and relationships that a particular user is interested in
- Different views may have different representation of the same data (e.g. US/UK/RO date representation)
- Universal relation: all information gathered from users
- Roles: Database Designer, Data Administrator

Conceptual level (schema)

- **The community view of the database. This level describes what data is stored in the database and the relationships among data.**

- Complete view of data requirements of an organization, containing:
  - All entities, their attributes, and their relationships
  - Constraints on the data
  - Semantic information about the data
  - Security and integrity information

- Does not contain any storage-dependent details

- Roles: Database Administrator, Data Administrator (security, privacy), Database Designer

Internal level (schema)

- **The physical representation of the database , an implementation of conceptual level. This level describes how the data is stored in the database: tables, indexes, sequences, views etc.**

- It covers the physical implementation (data structures, file organizations) of the database to achieve optimal runtime performance and storage space utilization, such as:
  - Storage space allocation for data and indexes
  - Record descriptions for storage
  - Record placement
  - Data compression and data encryption techniques

- This is the interface with the operating system

- Below this level there is a physical level managed by OS.

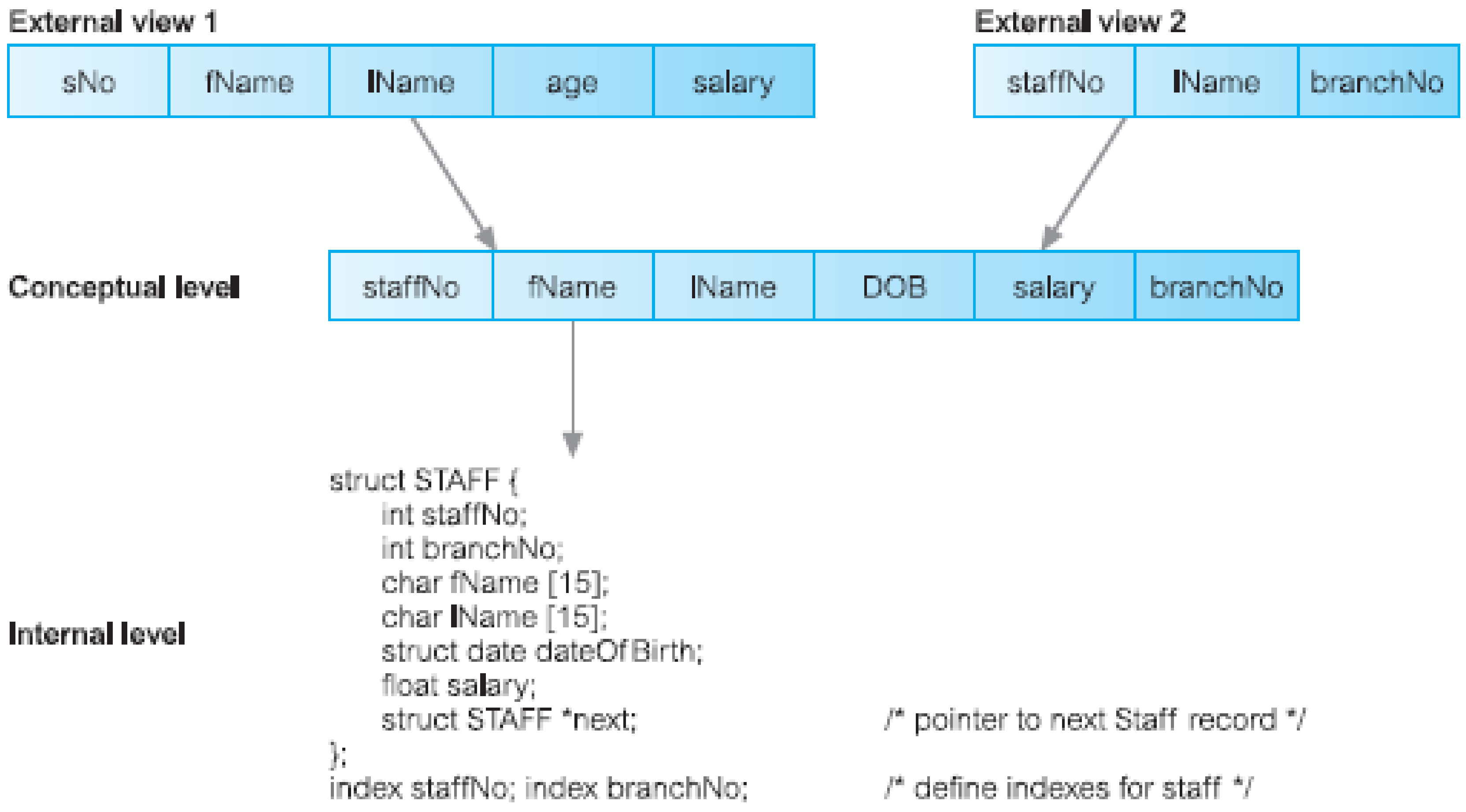
Schemas, Mappings and Instances

- Schemas: external (multiple) / conceptual (unique) / internal (unique)

- Conceptual/internal mapping - enables a DBMS to find a record or combination of records in physical storage that forms a logical record in conceptual model.

- External/conceptual mapping – enables a DBMS to map names in the user’s view onto the relevant parts of the conceptual schema

Schemas, Mappings and Instances



```
struct STAFF {
        int staffNo;
        int branchNo;
        char fName [15];
        char lName [15];
        struct date dateOfBirth;
        float salary;
        struct STAFF *next;                /* pointer to next Staff record */
};
index staffNo; index branchNo;             /* define indexes for staff */
```

Diagram taken from *Database Systems - A Practical Approach to Design, Implementation, and Management (4th edition)* by Thomas Connolly and Carolyn Begg, Addison-Wesley, 2004

Data Independence

- Upper levels are not impacted by changes in lower levels (ANSI / SPARC architecture)

- **Logical data independence** – refers to the immunity of external schemas to changes in the conceptual schema (addition or removal of new entities, attributes or relationships)

- **Physical data independence** – refers to the immunity of the conceptual schema to changes in the internal schema (using different file organizations or storage structures, different storage devices)

Database Design Theory

We shall focus on the Conceptual level for the remainder of this chapter

Running example

Let’s consider the table below; this report has been provided to IT Dept by the Student Help Center of our university.

CNP	Student Name	Course Name	Major	Faculty	Hobbies
1234567890 012	Ionescu Andrei	Databases I	CS	FMI	surfing, skiing
1234567890 012	Ionescu Andrei	Algebra	CS	FMI	football
1114567890 012	Popa Alexandra	Databases I	MATH	FMI	cooking
1114567890 034	Ionescu Andrei	History of British Art	PAINTING	FAD	volleyball

Modification anomalies

Anomalies of this design:

- **Redundancy** (CNP is associated to a particular student many times)
- **Update anomaly**: Updating one fact in a relation requires us to update multiple tuples => update facts differently in different places (not all tuples are correctly updated) => inconsistencies (Ex: update the name of student with a specified CNP)
- **Deletion anomaly:** Deleting one fact or data point from a relation results in other information being lost. Ex: deleting all tuples with hobby Biking -> it will delete all students and we will lose all information about those students; but, if a student has several hobbies, he will still remain in the database
- **Insertion Anomaly**: Inserting a new fact or tuple into a relation requires we have information from two or more entities – this situation might not be feasible. Ex: in order to insert a new class enrolment we need to supply the major and faculty, although this information may already be available in the database.

Running example

Students enrollment – modified version

~~**Enrollments**(CNP, StudentName, CourseName, Major, Faculty, Hobby)~~

Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)

Enrollments(CNP, CourseName, Dept, Date)

Running example

It seems to be better, but

- How good is it? Should we further decompose the tables?
- What anomalies still manifests?
- Lack of a formal background
- Lack of an algorithm for decomposition (repeatability)

Database Design Theory

- set a formal framework for database design
- useful to assess the quality of database design

Normalization (Design by decomposition)

Relational design by decomposition (Role: Database designer)

- Initially “mega” relations and properties of data we are storing
- Decomposition of these mega relations based on properties (semantics)
- Results a new set of relations that satisfies some *normal forms* (i.e. no anomalies, no data is lost)

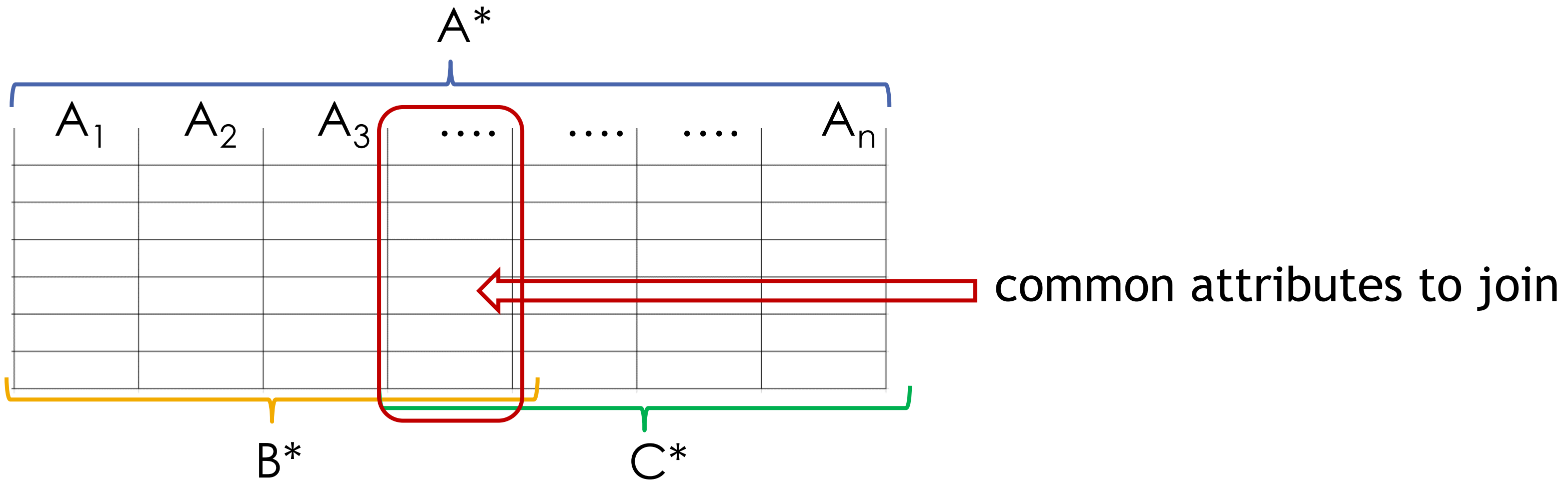
Lossless join decomposition of a relation

Let $R(A_1, A_2, ..., A_n)$, $R1(B_1, ..., B_k)$ and $R2(C_1, ..., C_m)$ be three relation. We note $A^* = \{A_1, A_2, ..., A_n\}$, $B^* = \{B_1, ..., B_k\}$ and $C^*=\{C_1, ..., C_m\}$ the set of attributes of three relations, respectively.

DEF: R1 and R2 are a lossless join decomposition of R iff:

$A^* = B^* \cup C^*$ and $R1 \bowtie R2 = R$ ($\bowtie$ - natural join)

$R1 = \Pi_{B^*}(R)$ and $R2 = \Pi_{C^*}(R)$



Dependencies

- Functional dependencies

- Multivalued dependencies

Functional dependencies

A Functional Dependency (FD) describes a relationship between the *attributes* within a single relation.

DEF: Given a relation R, an attribute B is *functionally dependent* on another attribute A (and we write A -> B) if we can use the value of attribute A to determine the value of B. We also say that “A determines B”

DEF: Formally, given R a relation, A and B attributes of R, and t, u tuples, then A -> B iff:

$$\forall t, u \in R \colon t.A = u.A \Rightarrow t.B = u.B$$

i.e. if two tuples agree on values of A they will agree on values of B as well
i.e. if values of A attribute are the same on one tuple, values for B will be the same as well.

Functional dependencies

DEF (Generalization for set of attributes): Given $\bar{A} = \{A_1, A_2, \ldots, A_j\}$ and $\overline{B} = \{B_1, B_2, \ldots, B_k\}$ two attributes sets of R, $\bar{A}$ determines $\overline{B}$ ($\bar{A} \rightarrow \overline{B}$) iff:

$$\forall t, u \in R\colon\ t.[A_1, \ldots, A_j] = u.[A_1, \ldots, A_j] \\ \Rightarrow\ t.[B_1, \ldots, B_k] = u.[B_1, \ldots, B_k]$$

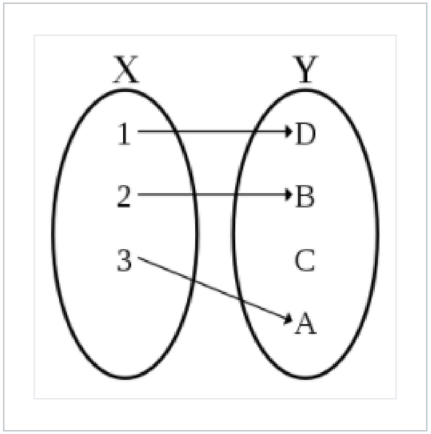
DEF: $A_1, A_2, \ldots, A_j$ are called **determinants.**

Remarks:

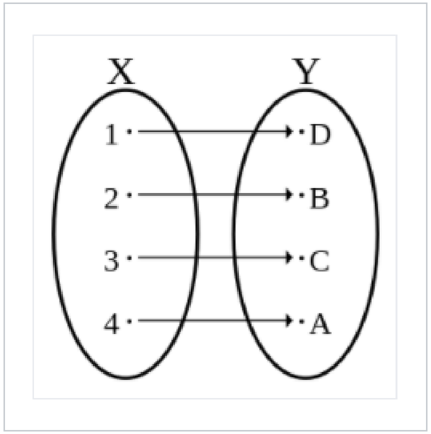
- A FD is a knowledge of a real world that is being captured in our model.

- All tuples of a relation must adhere to all FDs.

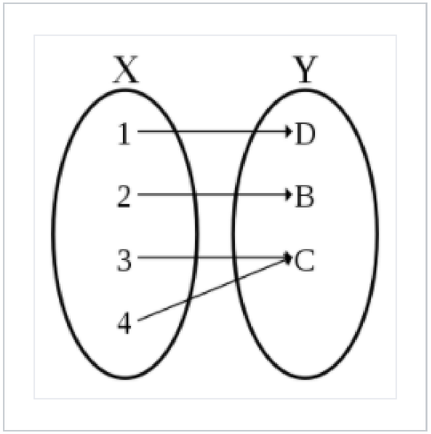
Functional dependencies



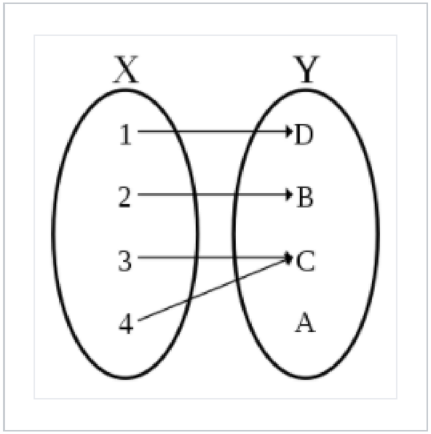
An injective non-surjective function (injection, not a bijection)



An injective surjective function (bijection)



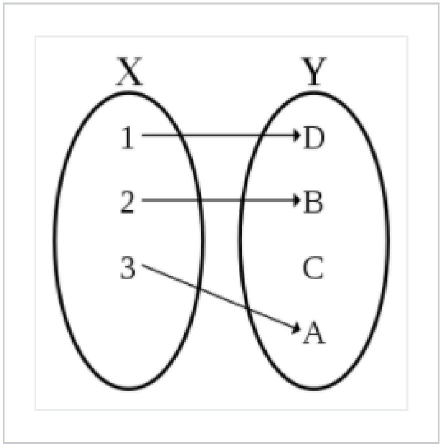
A non-injective surjective function (surjection, not a bijection)



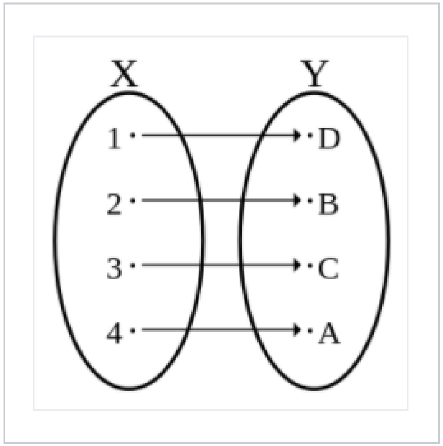
A non-injective non-surjective function (also not a bijection)

Source: https://en.wikipedia.org/wiki/Injective_function

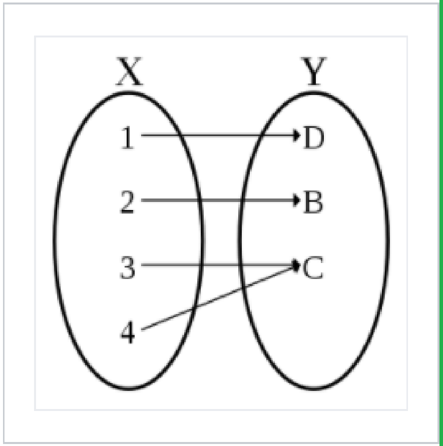
Functional dependencies



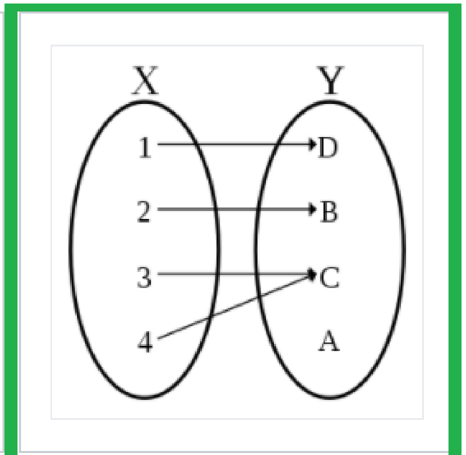
An injective non-surjective function (injection, not a bijection)



An injective surjective function (bijection)



A non-injective surjective function (surjection, not a bijection)



A non-injective non-surjective function (also not a bijection)

Functional Dependency

Source: https://en.wikipedia.org/wiki/Injective_function

Functional dependencies. Example

Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)

Enrollments(CNP, CourseName, Dept, Date)

Suppose that student’s priority is determined by his/her TotalCredits as follows:

50 <= TotalCredits then Priority = 1

40 <= TotalCredits < 50 then Priority = 2

TotalCredits < 40 then Priority = 3

Based on this relationship we can say that “two tuples with same TotalCredits have same Priority” and write this as a FD: **TotalCredits -> Priority**

Find other FD in Student and Enrollment relations.

Functional dependencies

DEF: $\bar{A} \rightarrow \bar{B}$ is trivial if $\bar{B} \subset \bar{A}$

DEF: $\bar{A} \rightarrow \bar{B}$ is non-trivial if $\bar{B} \not\subset \bar{A}$

DEF: $\bar{A} \rightarrow \bar{B}$ is completely non-trivial if $\bar{B} \cap \bar{A}$ is empty.

Armstrong's axioms

Reflexivity:

if B ⊆ A then $\bar{A} \rightarrow B$

Augmentation:

if $\bar{A} \rightarrow \bar{B}$ then $\overline{AC} \rightarrow \overline{BC}$ for any C

Transitivity:

if $\bar{A} \rightarrow \bar{B}$ and $\bar{B} \rightarrow \bar{C}$ then $\bar{A} \rightarrow \bar{C}$

Pseudo-transitivity

if $\bar{A} \rightarrow \bar{B}$ and $\overline{BD} \rightarrow \bar{C}$ then $\bar{A}D \rightarrow \bar{C}$ for any D

Note: Transitivity is a special case of pseudo-transitivity when D is null.

Rules derived from Armstrong's axioms

Splitting (decomposition) rule:

if $\bar{A} \rightarrow \{B_1, B_2, \ldots, B_n\}$ then $\bar{A} \rightarrow B_1$, $\bar{A} \rightarrow B_2$, …, $\bar{A} \rightarrow B_n$

Combining (union) rule:

if $\bar{A} \rightarrow B_1$, $\bar{A} \rightarrow B_2$, …, $\bar{A} \rightarrow B_n$ then $\bar{A} \rightarrow \{B_1, B_2, \ldots, B_n\}$

Question: If $\{A_1, A_2, \ldots, A_n\} \rightarrow B$ then $A_1 \rightarrow B$, $A_2 \rightarrow B$, …, $A_n \rightarrow B$ is it true?

Time for a Quiz

Attributes Set Closure

DEF [Closure of attributes] Given R, a set of FDs and $\bar{A}$ = a set of attributes from R. The closure of $\bar{A}$ ($\bar{A}^+$) is the set of all attributes B such that $\bar{A} \rightarrow B$, i.e. all attributes functionally determined by the set $\bar{A}$.

Algorithm to compute the closure:

1. start with the set $\bar{A}^+ = \bar{A} = \{A_1, A_2, \dots, A_n\}$
2. repeat until no change

    if $\bar{X} \rightarrow \bar{B}$ and $\bar{X}$ is in the closure $\bar{A}^+$ then add $\bar{B}$ to $\bar{A}^+$

Remark: A subset $\bar{A}$ functionally determines another subset $\bar{B}$ if $\bar{B} \subset \bar{A}+$

Exercise

Compute the closure {CNP, MajorCode}$^+$ for relation Student.

Exercise

Compute the closure {CNP, MajorCode}$^+$ for relation Students.

Answer: {CNP, MajorCode}$^+$ = Students*

Keys

DEF: [Superkey] If $\bar{A}^{+}$ is the set of all attributes of a relation R (R*) then $\bar{A}$ is a superkey in R.

How can we find the candidate keys (= irreducible superkey) given all FD?

Keys

DEF: [Superkey] If $\bar{A}^+$ is the set of all attributes of a relation R (R*) then $\bar{A}$ is a superkey in R.

How can we find the candidate keys (= irreducible superkey) given all FD?

Algorithm to compute the candidate keys of a relation:

1. Consider every subset $\bar{A}$ of R in increasing size (first $\bar{A}$ is the set composed of each attribute, then consider 2-attribute subsets etc.)
2. Compute set's closure, i.e. $\bar{A}+$
3. If $\bar{A}+$ = R* then $\bar{A}$ is a candidate key
4. If $\bar{A}$ is the last subset of its cardinality class (i.e. number of attributes in the set) then stop; otherwise go to 1

Functional dependencies and keys

Remark: Given a relation R, for any key K or R, K $\rightarrow$ R* (all other attributes). Thus, FD are generalizations of keys because any key functionally determines all other attributes

Remark: Not all determinants of a FD are necessarily keys (e.g. TotalCredits -> Priority and TotalCredits is not a key)

DEF: [non-prime attribute] A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation.

Reasoning with FD

DEF: [Follows from] If S_1 and S_2 are two sets of FDs, S_2 follows from S_1 if every instance satisfying S_1 also satisfies S_2.

Example: For Students relation, if

S_1 = {CNP->TotalCredits, TotalCredits->Priority},

S_2 = {CNP->Priority}

then S_2 follows from S_1

Q: How to test whether a FD $\bar{A} \rightarrow \bar{B}$ follows from a given set of functional dependencies S?

A: Compute $\bar{A}^+$ using only the FDs in S

If $\bar{B}$ is a subset of $\bar{A}^+$ then $\bar{A} \rightarrow \bar{B}$ follows from S

Non-key FD’s

DEF: A non-trivial FD $\bar{A} \rightarrow$ B where $\bar{A}$ is not a super key is called non-key FD.

Since $\bar{A}$ is not a super key, there are some attributes (say C) that are not functionally determined by $\bar{A}$.

Non-key FD cause:

- redundancy,
- update anomaly,
- deletion anomaly

Functional dependencies

The Holly Grail is to find the *minimal* set of completely *non-trivial FD* such that all FD’s that hold on the relation *follow from* the FD in this set.

FD are useful for:

- Relational design by decomposition (FD => BCNF)
- Data storage (compression)
- Query optimizations

Exercise

Consider relation R(A, B, C, D, E) with the following functional dependencies: $D \rightarrow C$, $CE \rightarrow A$, $D \rightarrow A$, $AE \rightarrow D$. Which of the following attribute sets is a superkey of R?

a) {D}
b) {A, B}
c) {A, B, E}
d) {C, D, E}

Exercise

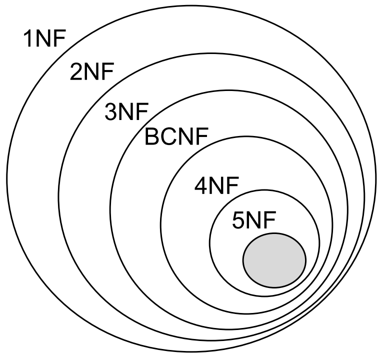
Consider relation R(A, B, C, D, E) with the following functional dependencies: $D \to C$, $CE \to A$, $D \to A$, $AE \to D$. Which of the following attribute sets is a superkey of R?

a) {D}
b) {A, B}
c) **{A, B, E}**
d) {C, D, E}

Normal Forms (NF)

Normal Form: A class of relations which are free from a certain set of modification anomalies.

- First normal form (1NF)
- Second normal form (2NF)
- Third normal form (3NF)
- Boyce-Codd normal form (BCNF)
- Fourth normal form (4NF)
- Fifth normal form (5NF)



These forms are cumulative. A relation in 3NF is also in 2NF and 1NF.

Normalization Process

The *Normalization Process* for a given relation consists of:

1. Compute the *functional dependencies (FD)* of the relation. (Remark: Sample data (tuples) for the relation can assist with this step.)
2. Compute the *candidate keys* of the relation
3. Apply the definition of each normal form (starting with 1NF).
4. If a relation fails to meet the definition of a normal form, change the relation (most often by decomposing the relation into two new relations) until it meets the definition.
5. Re-test the modified/new relations to ensure they meet the definitions of each normal form.

First Normal Form (1NF)

DEF: A relation is in first normal form if it meets the definition of a relation.

Definition of a relation:

1. Each attribute (column) value must be a single value only.
2. All values for a given attribute (column) must be of the same type (domain).
3. Each attribute (column) name must be unique.
4. The order of attributes (columns) is insignificant
5. No two tuples (rows) in a relation can be identical.
6. The order of the tuples (rows) is insignificant.

If you have a *key* defined for the relation, then you can meet the *unique row* requirement.

Normalization to 1NF

Example 1: **Enrollments**(CNP, StudentName, CourseName, Major, Faculty, Hobby)

1900101110011, Andrei, …., {Biking, Soccer}
1910104150011, Elena, …., {Reading, Biking}

TO 1NF

1900101110011, Andrei, …., Biking
1900101110011, Andrei, …., Soccer
1910104150011, Elena, …., Reading
1910104150011, Elena, …., Biking

Compare the keys of the **original** and **modified** relations.

Example 2: **Employees**(ID, Name, Position)

1, Ionescu, {Designer, Programmer}
2, Vasile, {Accountant, Economist}

Running example – revised version

~~**Enrollments**(CNP, StudentName, CourseName, Major, Faculty, Hobby)~~

Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)

Enrollments(CNP, CourseName, Dept, Date)

Second Normal Form (2NF)

DEF: A relation is in second normal form (2NF) if it is in 1NF and it does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation.

- Another way to say this: A relation is in 2NF if it is free from partial-key dependencies against any of the candidate keys.

Examples:

- Students(<u>CNP</u>, Name, Address, <u>MajorCode</u>, MajorName, Faculty, TotalCredits, Priority)

- Stocks(Company, <u>Symbol</u>, Headquarters, <u>Date</u>, ClosePrice)

Second Normal Form (2NF)

DEF: A relation is in second normal form (2NF) if it is in 1NF and it does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation.

- Another way to say this: A relation is in 2NF if it is free from partial-key dependencies against any of the candidate keys (CK).

Examples:

- Students(<u>CNP</u>, Name, Address, <u>MajorCode</u>, MajorName, Faculty, TotalCredits, Priority)
  - not in 2NF because CNP -> {Name, Address} and the candidate key is {CNP, MajorCode}

- Stocks(Company, <u>Symbol</u>, Headquarters, <u>Date</u>, Close_Price)
  - not in 2NF because Symbol -> {Company, Headquarters} and the CK is {Symbol, Date}

Normalization to 2NF

- List all FD
- Test all FD against all CKs to all discover violations (i.e., partial key dependencies)
- If **A1 -> X** is a partial key dependency then decompose the original relation in 2 relations (using lossless join decomposition):
    - R(A1, A2, X, Y) => R1(A1, A2, Y) and **R2(A1, X**)
    - Check for 1NF and 2NF compliance of new relations

Normalization to 2NF - Example

- Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)

- Using PKD: MajorCode -> {MajorName, Faculty} decompose Students in:
  - =>Students1(CNP, Name, Address, MajorCode, TotalCredits, Priority) – not in 2NF
  - **=>Students2(MajorCode, MajorName, Faculty) - in 2NF**

- Using PKD: CNP -> {Name, Address, TotalCredits, Priority} decompose Students1 in:
  - **=> Students1_1(CNP, MajorCode) – in 2NF**
  - **=> Students1_2(CNP, Name, Address, TotalCredits, Priority) - in 2NF**

Second Normal Form (2NF). Discussion

- Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)

- What if I add a surrogate candidate key, ID, to Students table, which becomes Students(ID, CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority), is it in 2NF now?

Second Normal Form (2NF). Discussion

- Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)

- What if I add a surrogate candidate key, ID, to Students table, which becomes Students(ID, CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority), is it in 2NF now?
  - A: **NO,** because {CNP, MajorCode} is still a CK and all previous violations still stands!

- Relations that have only one single-attribute CK are automatically in 2NF. This is one of the reasons why artificial identifiers, i.e. surrogate keys, are used as candidate/primary keys.

- Reference: https://en.wikipedia.org/wiki/Second_normal_form

Third Normal Form (3NF)

DEF: A relation is in third normal form (3NF) it is in 2NF and every non-prime attribute is non-transitively dependent on every key of the relation.

DEF: Given relation R(A, … B, …, C), where A, B and C are three attributes. A, B and C are in a transitive dependency if A->B and B->C then A->C.

Examples:
Students(<u>CNP</u>, Name, Address, TotalCredits, Priority)

Companies(Company, <u>Symbol</u>, Headquarters)

Third Normal Form (3NF)

DEF: A relation is in third normal form (3NF) it is in 2NF and every non-prime attribute is non-transitively dependent on every key of the relation.

DEF: Given relation R(A, … B, …, C), where A, B and C are three attributes. A, B and C are in a transitive dependency if A->B and B->C then A->C.

Examples:
Students(CNP, Name, Address, TotalCredits, Priority)

- not in 3NF, CNP -> TotalCredits and TotalCredits -> Priority

Companies(Company, Symbol, Headquarters)

- not in 3NF, Symbol -> Company and Company -> Headquarters

Normalization to 3NF

- List all FD
- Test all FD to discover any transitive dependencies
- If K -> X1 and **X1 -> X2** is a transitive dependency then decompose the original relation R in 2 relations using lossless join decomposition
    - R(K, X1, X2, X3) => R1 (K,X1,X3) si **R2 (X1, X2)**
- Check for 1NF, 2NF and 3NF compliance of new relations

Normalization to 3NF - Example

- **Example:** Students1_2 (<u>CNP</u>, Name, Address, TotalCredits, Priority)
  - FD: TotalCredits->Priority
  - => **Students1_2_1(<u>CNP,</u> Name, Address, TotalCredits) – in 3 NF**
  - => **Students1_2_2(<u>TotalCredits</u>, Priority) – in 3 NF**

Second Normal Form (3NF). Discussion

- Every non-key attribute must provide a fact about the key, the whole key, and nothing but the key (Bill Kent, 1983)
- => a relation is in 3NF if all the attributes are functionally dependent on solely the primary key.
- Most the 3NF relations are free of update, insertion, and deletion anomalies. Certain types of 3NF tables, rarely met with in practice, are affected by such anomalies. (will see an example later)
- Reference: https://en.wikipedia.org/wiki/Third_normal_form

Boyce-Codd Normal Form (BCNF / 3.5NF)

DEF: A relation R is in Boyce-Codd normal form (BCNF) iff for every FD $\bar{A} \rightarrow \bar{B}$, at least one of the following holds

- $\bar{A} \rightarrow \bar{B}$ is a trivial FD
- $\bar{A}$ is a superkey of R

- Another way to say it: every determinant $\bar{A}$ of a FD $\bar{A} \rightarrow \bar{B}$ is a candidate key

- If a relation is in BCNF then all redundancy based has been remove.

BCNF separates a relation so that we capture each piece of information exactly once.

Boyce-Codd Normal Form (BCNF / 3.5NF)

DEF: A relation R is in Boyce-Codd normal form (BCNF) iff for every FD $\bar{A} \rightarrow \bar{B}$, at least one of the following holds

- $\bar{A} \rightarrow \bar{B}$ is a trivial FD
- $\bar{A}$ is a superkey of R

Examples:

- Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)
- Students(CNP, Name, Address, TotalCredits, Priority)
- Enrollments(CNP, CourseName, Dept, Date)

Normalization to BCNF

- Given R
- List all of determinants (Ā) of FD (**Ā -> B**)
- See if each determinant (Ā) can act as a CK, by computing $\bar{A}^+$
- For any determinant that is <u>not</u> a CK, create a new relation R1 from the FD. Retain the determinant in the original relation.
    - R(Ā, B, rest) => **R1(Ā, B)** and R2(Ā, rest)

Normalization to BCNF. Example

Students(CNP, Name, Address, MajorCode, MajorName, Faculty, TotalCredits, Priority)

FD1: CNP -> {Name, Address, TotalCredits, Priority},

FD2: MajorCode -> {MajorName, Faculty},

FD3: TotalCredits -> Priority

- Using FD2

⇒ S1(CNP, Name, Address, MajorCode, TotalCredits, Priority)

⇒ S2(MajorCode, MajorName, Faculty)

- Using FD3 to normalize S1

⇒ S1_1(TotalCredits, Priority)

⇒ S1_2(CNP, Name, Address, MajorCode, TotalCredits)

- Using FD1 to normalize S1_2

⇒ S1_2_1(CNP, Name, Address, TotalCredits)

⇒ S1_2_2(CNP, MajorCode)

Rename relations S2, S1_1, S1_2_1 and S1_2_2 to something meaningful…

BCNF vs. 3NF Discussion

A 3NF table that does not have multiple overlapping candidate keys is guaranteed to be in BCNF. Depending on what its functional dependencies are, a 3NF table with two or more overlapping candidate keys may or may not be in BCNF.

Person	ShopType	NearestShop
Davidson	Optician	Eagle Eye
Davidson	Hairdresser	Snippets
Wright	Bookshop	Merlin Books
Fuller	Bakery	Doughy's
Fuller	Hairdresser	Sweeney Todd's
Fuller	Optician	Eagle Eye

BCNF vs. 3NF Discussion

Person	ShopType	NearestShop
Davidson	Optician	Eagle Eye
Davidson	Hairdresser	Snippets
Wright	Bookshop	Merlin Books
Fuller	Bakery	Doughy's
Fuller	Hairdresser	Sweeney Todd's
Fuller	Optician	Eagle Eye

- FDs: {Person, ShopType} -> NearestShop, NearestShop -> ShopType (assuming one shop has only one shopping type)
- Candidate keys: {Person, ShopType}, {Person, NearestShop}
- The table is in 3NF (no transitive dependencies) but not in BCNF because ShopType is dependent on a non-superkey (NearestShop)
- What anomalies may arise?
- BCNF decomposition
  - R1(Person, NearestShop)
  - R2(NearestShop, ShopType)

Reference: https://en.wikipedia.org/wiki/Boyce%E2%80%93Codd_normal_form

Multivalued dependencies

DEF: Multivalued Dependency: A type of functional dependency where the determinant can determine more than one value.

In other words, given R(A, B, C, ….) a relation with at least three attributes A, B and C all the following holds:

- Given A, one can determine multiple values of B.
- Given A, one can determine multiple values of C.
- B and C are independent of one another.

Multivalued dependencies

Examples:

1. **Enrollments**(CNP, CourseName, Hobby)

- MVD1: A student may enroll into one or more courses (CNP ->> CourseName)
- MVD2: A student may have one or more hobbies (CNP ->> Hobby)
- College and Hobby are independent

2. **Students**(CNP, Name, Address, MajorCode, TotalCredits)

- MVD1: A student may study one or more majors (CNP ->> MajorCode)
- MVD2: A student may have one or more addresses (CNP ->> Address)
- Major and address are totally unrelated

Multivalued dependencies

Restaurant	Pizza Variety	Delivery Area
A1 Pizza	Thick Crust	Springfield
A1 Pizza	Thick Crust	Shelbyville
A1 Pizza	Thick Crust	Capital City
A1 Pizza	Stuffed Crust	Springfield
A1 Pizza	Stuffed Crust	Shelbyville
A1 Pizza	Stuffed Crust	Capital City
Elite Pizza	Thin Crust	Capital City
Elite Pizza	Stuffed Crust	Capital City
Vincenzo's Pizza	Thick Crust	Springfield
Vincenzo's Pizza	Thick Crust	Shelbyville
Vincenzo's Pizza	Thin Crust	Springfield
Vincenzo's Pizza	Thin Crust	Shelbyville

{Restaurant} ->> {Pizza Variety}

{Restaurant} ->> {Delivery Area}

Multivalued dependencies

Enrollments(CNP, CourseName, Hobby)

- MVD1: A student may enroll into one or more courses (CNP ->> CourseName)
- MVD2: A student may have one or more hobbies (CNP ->> Hobby)
- College and Hobby are independent
- FD: none
- Candidate key: {CNP, CourseName, Hobby}
- It is in BCNF
- Is it a good design? Anomalies:
  - Removing an enrollment will also remove hobbies
  - Cannot add hobbies without enrollments
  - etc.

Multivalued dependencies

DEF: Given $\bar{A}$ = {A$_1$, A$_2$, …, A$_j$} and $\bar{B}$ = {B$_1$, B$_2$, …, B$_k$} two attribute sets of R, $\bar{A}$ multivalued determines $\bar{B}$ ($\bar{A} \twoheadrightarrow \bar{B}$) iff:

$$\forall t, u \in R: \; t.[\bar{A}] = u.[\bar{A}] \; then \; \exists v \in R:$$

$$v[\bar{A}] = t.[\bar{A}] and \; v.[\bar{B}] = t.[\bar{B}] and \; v.[rest] = u.[rest]$$

	$\bar{A}$	$\bar{B}$	$rest$
t	$\bar{a}$	$\bar{b}_1$	$\bar{r}_1$
u	$\bar{a}$	$\bar{b}_2$	$\bar{r}_2$
v	$\bar{a}$	$\bar{b}_1$	$\bar{r}_2$
w	$\bar{a}$	$\bar{b}_2$	$\bar{r}_1$
	:	:	:

Multivalued dependencies

DEF: The MVD Ā —» B is trivial iff B ⊆ Ā **or Ā U B = R*.**

DEF: The MVD Ā —» B is non-trivial if it is a MVD, which is not trivial.

Lemma: Each FD is a MVD: if Ā -> B then Ā ->> B.

Demo:

	$\bar{A}$	$\bar{B}$	$rest$
t	$\bar{a}$	$\bar{b}_1$	$\bar{r}_1$
u	$\bar{a}$	$\bar{b}_2$	$\bar{r}_2$
v	$\bar{a}$	$\bar{b}_1$	$\bar{r}_2$
	⋮	⋮	⋮
	⋮	⋮	⋮

Complete FD and MVD rules

- FD reflexivity, augmentation, and transitivity

- MVD complementation:

If A ->> B, then A ->> attrs(R) – A – B

- MVD augmentation:

If A ->> B and V ⊆ W, then AW ->> BV

- MVD transitivity:

If A ->> B and B ->> Z, then A ->> Z – B

- Replication (FD is MVD):

If A → B, then A ->> B

- Coalescence:

If A ->> B and Z ⊆ B and there is some W disjoint from B such that W → Z, then A → Z

Fourth Normal Form (4NF)

DEF: A relation is in fourth normal form (4NF) iff for every non-trivial MVD $\bar{A}$ ->> B, $\bar{A}$ is a superkey (either a CK or a superset thereof).

Normalization to 4NF

- Let $\bar{A} \twoheadrightarrow B$ be a non-trivial MVD
- Split R in two new relations **R1($\bar{A}$, B)** and R2($\bar{A}$, rest)
- Apply the decomposition algorithm for R1 and R2

Normalization to 4NF. Example

1. **Enrollments**(CNP, CourseName, Hobby)

$\Rightarrow$ S1(CNP, CourseName)

$\Rightarrow$ S2(CNP, Hobby)

2. Students(CNP, Name, Address, MajorCode, TotalCredits)

$\Rightarrow$ S1(CNP, Name, Address)

$\Rightarrow$ S2(CNP, MajorCode, TotalCredits)

Normalization to 4NF. Discussion

- In a study of 40 organizational databases, over 20% contained one or more tables that violated 4NF while meeting all lower normal forms [Wu, M. S. et al – 1992]

- Only in rare situations does a 4NF table not conform to the higher normal form 5NF. These are situations in which a complex real-world constraint governing the valid combinations of attribute values in the 4NF table is not implicit in the structure of that table.

References: https://en.wikipedia.org/wiki/Fourth_normal_form

Fifth Normal Form (5NF)

- DEF: A relation is in fifth normal form (5NF) iff every non-trivial join dependency in that table is implied by the candidate keys.

- DEF: A join dependency {A, B, … Z} on R is implied by the candidate key(s) of R if and only if each of A, B, …, Z is a superkey for R

- DEF: A relation is in fifth normal form (5NF) if it cannot be split in 2 or more relations so that the join uniquely reconstructs the original relation (a relation cannot be reassembled back to original form)

Fifth Normal Form (5NF). Example

Traveling Salesman Product Availability By Brand

Traveling Salesman	Brand	Product Type
Jack Schneider	Acme	Vacuum Cleaner
Jack Schneider	Acme	Breadbox
Mary Jones	Robusto	Pruning Shears
Mary Jones	Robusto	Vacuum Cleaner
Mary Jones	Robusto	Breadbox
Mary Jones	Robusto	Umbrella Stand
Louis Ferguson	Robusto	Vacuum Cleaner
Louis Ferguson	Robusto	Telescope
Louis Ferguson	Acme	Vacuum Cleaner
Louis Ferguson	Acme	Lava Lamp
Louis Ferguson	Nimbus	Tie Rack

Products of the type designated by Product Type, made by the brand designated by Brand, are available from the traveling salesman designated by Traveling Salesman.

Fifth Normal Form (5NF). Example

Traveling Salesman Product Availability By Brand

Traveling Salesman	Brand	Product Type
Jack Schneider	Acme	Vacuum Cleaner
Jack Schneider	Acme	Breadbox
Mary Jones	Robusto	Pruning Shears
Mary Jones	Robusto	Vacuum Cleaner
Louis Ferguson	Robusto	Vacuum Cleaner
Louis Ferguson	Robusto	Telescope
…	…	…

- FD: none
- MVD: none
- CK/PK: {Travelling Salesman, Brand, Product Type}
- It is in 4NF

Fifth Normal Form (5NF). Example

Traveling Salesman Product Availability By Brand

If the following rule applies

> "A Traveling Salesman has certain Brands and certain Product Types in their repertoire. If Brand B1 and Brand B2 are in their repertoire, and Product Type P is in their repertoire, then (assuming Brand B1 and Brand B2 both make Product Type P), the Traveling Salesman must offer products of Product Type P those made by Brand B1 and those made by Brand B2."

then it is possible to split the table as follows:

- R1(Travelling Salesman, Product Type)
- R2(Travelling Salesman, Brand)
- R3(Brand, Product Type)

Fifth Normal Form (5NF). Discussion

- Only in rare situations does a 4NF table not conform to 5NF. These are situations in which a complex real-world constraint governing the valid combinations of attribute values in the 4NF table is not implicit in the structure of that table.
- If such a table is not normalized to 5NF, the burden of maintaining the logical consistency of the data within the table must be carried partly by the application responsible for insertions, deletions, and updates to it; and there is a heightened risk that the data within the table will become inconsistent. In contrast, the 5NF design excludes the possibility of such inconsistencies.

References: https://en.wikipedia.org/wiki/Fifth_normal_form

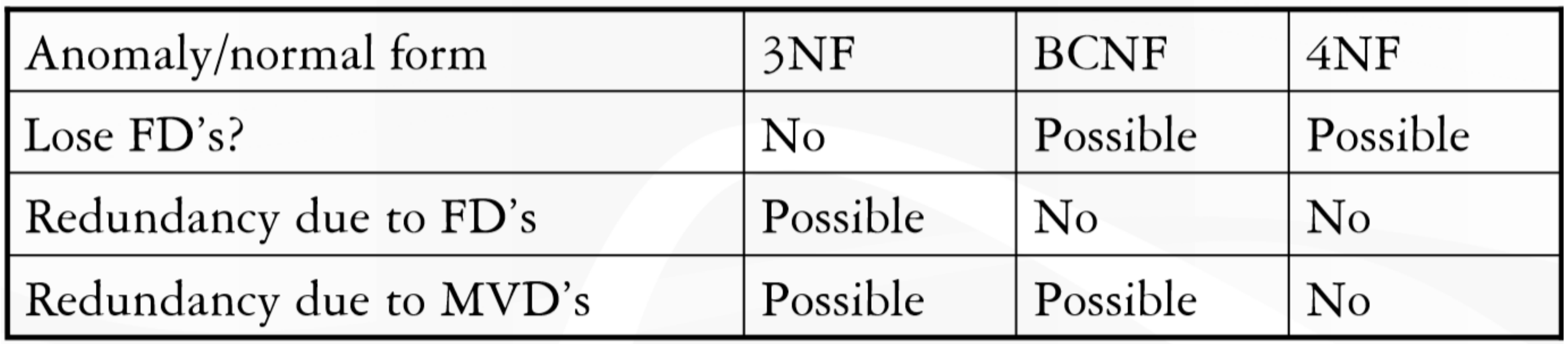
Final remarks

Normalization process transforms a dependency (partial key dependency, functional dependency, multivalued dependency transitive dependency) into a join relationship between two tables.

- PKD => 2NF
- TD => 3NF
- FD => BCNF
- MVD => 4NF

Attributes that are copied from one relation to another during decomposition become keys: PK in new relation and FK in the old relation.

3NF, BCNF, 4NF



Anomaly/normal form	3NF	BCNF	4NF
Lose FD's?	No	Possible	Possible
Redundancy due to FD's	Possible	No	No
Redundancy due to MVD's	Possible	Possible	No

De-Normalization

Shortcomings with normalized relations:

- Performance penalties
- Increased schema complexity

DEF: De-normalization: the process of re-assembly the original relations.

Examples:

1. Students relation in which we keep TotalCredits and Priority (for example, TotalCredits value may change after student issued his/her enrollment request)
2. Order_Line contains the Price of an item (due to contractual constraints)

Normalization recap

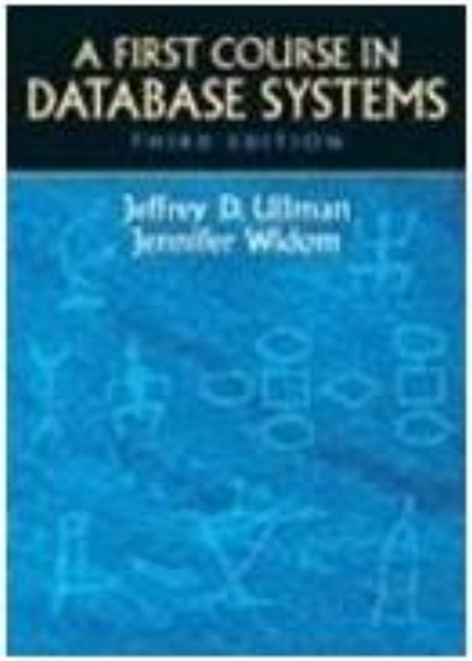
- Normalization is a *process* in which we systematically examine relations for *anomalies* and, when detected, remove those anomalies by splitting up the relation into two new, related, relations.

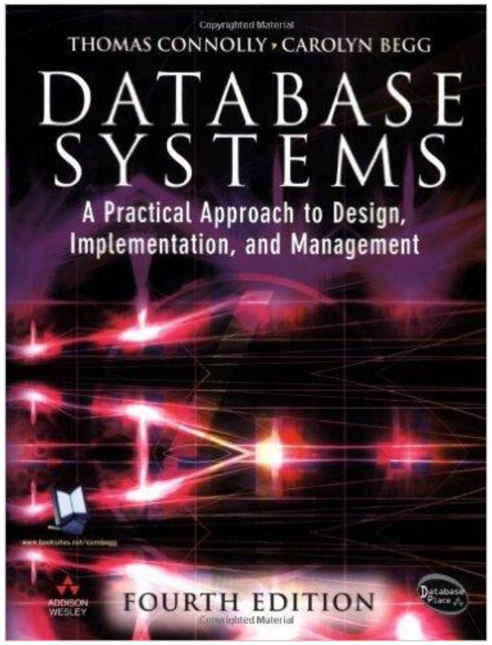
- Normalization is an important part of the database development process: during normalization, the database designers get their first real look into how the data are going to interact in the database.

- Finding problems with the database structure at this stage is strongly preferred to finding problems further along in the development process because at this point it is fairly easy to cycle back to the conceptual model and make changes.

- Normalization can also be thought of as a trade-off between data redundancy and performance. Normalizing a relation reduces data redundancy but introduces the need for joins when all of the data is required by an application such as a report query.

Bibliography (recommended)




THOMAS CONNOLLY • CAROLYN BEGG
DATABASE SYSTEMS
A Practical Approach to Design, Implementation, and Management
FOURTH EDITION
ADDISON WESLEY

Teoria generala a bazelor de date, I. Despi, G. Petrov, R. Reisz, A. Stepan, Mirton, 2000
Cap 6

A First Course in Database Systems (3rd edition) by Jeffrey Ullman and Jennifer Widom, Prentice Hall, 2007
Chapter 3

Database Systems - A Practical Approach to Design, Implementation, and Management (4th edition) by Thomas Connolly and Carolyn Begg, Addison-Wesley, 2004
Chapter 13 & 14

References

- Database Normalization
http://holowczak.com/database-normalization

- All-in-One Database Normalization Example
http://holowczak.com/database-normalization/12/