Relational Database Design

- One single, large table
- Simple ?
- Good ? or Bad? Or just Ugly?

Normal Forms

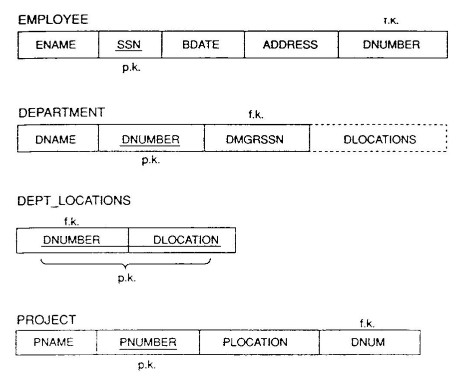
- We discussed how to fix 'bad' schemas
- but what is a 'good' schema?
- Informally: "we want tables where the attributes depend on the primary key, on the whole key, and nothing but the key"
- Formally: 'good', if it obeys a 'normal form'
- Typically: Boyce-Codd Normal form or the 3NF
- Normal forms are defined in terms of FDs

Functional dependency

- Definition: Let $R=(A_1, A_2,\ldots, A_n)$ and $X \subseteq R$ and $Y \subseteq R$
  $X \rightarrow Y$ if the value of X **uniquely** determines a value of Y
- A functional dependency is a property of the meaning or semantic of the attributes in a relation schema.
- We use our understanding of the semantics of the attributes of R – that is, how they relate to one another – to specify the FD that should hold an all relational instances.
- Functional dependence is a semantic notion.
  - Recognizing the FDs is part of the process of understanding what data means.

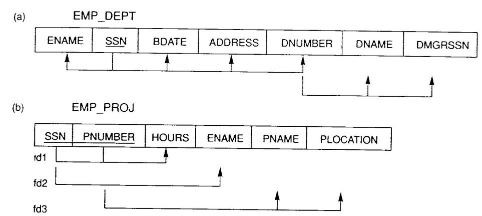
Good Database Schema

- Relations should have simple meaning



Bad Database Schema

- Relations should not have multiple meanings



Types of Functional Dependencies

- Trivial dependency: X→Y is *trivial* if it is true for any X and Y of any relation, regardless of X and Y semantics.
  - Ex1: A -> A
  - Ex2: If {A,B} a Key, then {A,B} → A (Y⊆X, X→Y is trivial)
- Partial dependency: X→Y is *partial* if there is an attribute A in X that can be removed from X and the dependency can still hold: X–{A} → Y
  - E.g., SUPPLY (SID,PID,DID, SCity, DCity, Qty)
    {SID, PID, DID} → Scity   SID → SCity
    {SID, PID, DID} → Dcity   DID → Dcity
- Full dependency: ??

Types of Functional Dependencies…

- Transitive dependency: X→Y is transitive in R if there is a set of attributes Z that is not a subset of any key of R and both X → Z and Z → Y hold
  - E.g., EMP (SSN, EName, DeptID, MGRSSN)
    (fd.1) SSN → DeptId
    (fd.2) DeptId → MGRSSN
    (from fd.1 & fd.2) SSN → MGRSSN
- Multivalued dependency : X→Y is multivalued dependency in R if X is a key, Z in R and Z→Y
  - E.g., DJP (DeptID, ProjectID, part)
    (fd.1) DeptId → part
    (fd.2) ProjectID → part

Non-Relational Table

- Unnormalized table/ NoSQL

MemID	Full Names	Address	Barcode	Movies Rented	Awards
1	Janet Jones	First Street Plot No 4	(0001, 0002)	(Pirates of the Caribbean, Clash of the Titans)	(15, 10)
2	Robert Phil	3rd Street 34	(0003, 0004)	(Forgetting Sarah Marsal, Daddy's Little Girls)	(16, 2)
3	Robert Phil	5th Avenue	0005	Clash of the titans	10

Order Lists/ Arrays

First Normal Form

- **1NF: First Normal Form**
  Every attribute has a single atomic value.
- Example in 1NF

MemID	Full Names	Address	Barcode	Movies Rented	Awards
1	Janet Jones	First Street Plot No 4	0001	Pirates of the Caribbean	15
1	Janet Jones	First Street Plot No 4	0002	Clash of the Titans	10
2	Robert Phil	3rd Street 34	0003	Forgetting Sarah Marsal	16
2	Robert Phil	3rd Street 34	0004	Daddy's Little Girls	2
3	Robert Phil	5th Avenue	0005	Clash of the Titans	10

- PK: MemID, Barcode
- FD1: MemID → (Full Names, Address)
- FD2: Barcode → Movies Rented
- FD3: Movies Rented → Awards

Not in Second Normal Form

- PK: MemID, Barcode
- FD1: MemID → (Full Names, Address)
- FD2: Barcode → Movies Rented
- FD3: Movies Rented → Awards

MemID	Full Names	Address	Barcode	Movies Rented	Awards
1	Janet Jones	First Street Plot No 4	0001	Pirates of the Caribbean	15
1	Janet Jones	First Street Plot No 4	0002	Clash of the Titans	10
2	Robert Phil	3rd Street 34	0003	Forgetting Sarah Marsal	16
2	Robert Phil	3rd Street 34	0004	Daddy's Little Girls	2
3	Robert Phil	5th Avenue	0005	Clash of the Titans	10

FD1 PK FD2 FD3

Second Normal Forms

- **2NF: Second Normal Form**
  It is in 1NF and does not have partial dependencies.
- Example in 2NF

PK: MemID

MemID	Full Names	Address
1	Janet Jones	First Street Plot No 4
2	Robert Phil	3rd Street 34
3	Robert Phil	5th Avenue

FD1:
MemID → (Full Names, Address)

PK: MemID, Barcode

MemID	Barcode	Movies Rented	Awards
1	0001	Pirates of the Caribbean	15
1	0002	Clash of the Titans	10
2	0003	Forgetting Sarah Marsal	16
2	0004	Daddy's Little Girls	2
3	0005	Clash of the Titans	10

FD2: Barcode → Movies Rented

FD3: Movies Rented → Awards

Third Normal Forms

- **3NF: Third Normal Form**
  It is in 2NF and does not have transitive dependencies to attributes that are not part of a key.
  - If X→A is an FD then (a) it is trivial, or (b) X is a superKey, or (c) A is a subset of a **candidate** Key.
- Counter example 3NF

PK: MemID, Barcode

FD2: Barcode → Movies Rented

FD3: Movies Rented → Awards

MemID	Barcode	Movies Rented	Awards
1	0001	Pirates of the Caribbean	15
1	0002	Clash of the Titans	10
2	0003	Forgetting Sarah Marsal	16
2	0004	Daddy's Little Girls	2
3	0005	Clash of the Titans	10

FD2 FD3

3NF Example (BCNF)

MemID	Full Names	Address
1	Janet Jones	First Street Plot No 4
2	Robert Phil	3rd Street 34
3	Robert Phil	5th Avenue

FD1:
MemID → (Full Names, Address)

MemID	Barcode
1	0001
1	0002
2	0003
2	0004
3	0005

PK:
MemID, Barcode

Barcode	Movies Rented
0001	Pirates of the Caribbean
0002	Clash of the Titans
0003	Forgetting Sarah Marsal
0004	Daddy's Little Girls
0005	Clash of the titans

FD2:
Barcode → Movies Rented

Movies Rented	Awards
Pirates of the Caribbean	15
Clash of the Titans	10
Forgetting Sarah Marsal	16
Daddy's Little Girls	2
Clash of the titans	10

FD3:
Movies Rented → Awards

3NF Example (Movies Rented→Barcode)

MemID	Full Names	Address
1	Janet Jones	First Street Plot No 4
2	Robert Phil	3rd Street 34
3	Robert Phil	5th Avenue

FD1:
MemID → (Full Names, Address)

MemID	Barcode
1	0001
1	0002
2	0003
2	0004
3	0005

PK:
MemID, Barcode

Barcode	Movies Rented	Awards
0001	Pirates of the Caribbean	15
0002	Clash of the Titans	10
0003	Forgetting Sarah Marsal	16
0004	Daddy's Little Girls	2
0005	Clash of the titans	10

FD2:
Barcode → Movies Rented

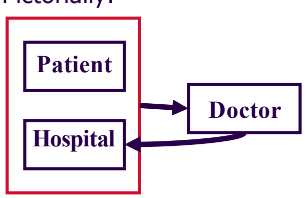
FD3:
Movies Rented → Awards

3NF Example

- Example: X→A and A is a subset of a **candidate** Key.

PATIENT-VISIT (<u>PATIENT,HOSPITAL</u>,DOCTOR)
fd.1 (Patient, Hospital) → Doctor
fd.2 Doctor → Hospital

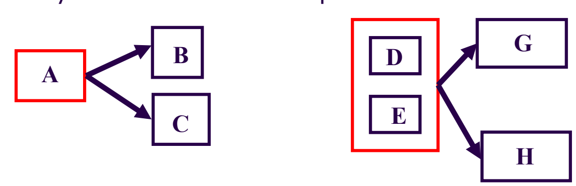
- Pictorially:



- It has a transitive dependency, but this is not to an attribute that are not part of a key.
- **Hospital** is a subset of a candidate Key (fd.1)

Normal Forms - BCNF

- **<u>Boyce-Codd Normal Form</u>: A relation is in 3NF and has no transitive dependencies**
  - if X → A is an FD, then
    (a) it is trivial, or (b) X is a superKey.
- Informally: everything depends on the **full key**, and nothing but the key
- Pictorially: we want a 'star' shape

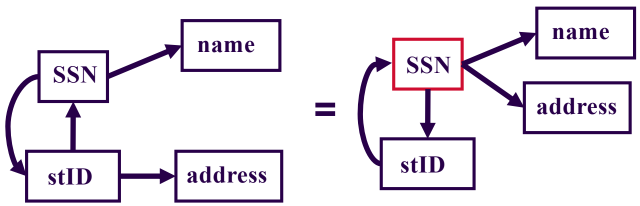


Normal Forms - BCNF

- or a star-like: (e.g., 2 candidate keys):

  STUDENT(SSN, stID, name, address)

  { fd.1 SSN → name, fd.2 SSN → stID,
  fd.3 stID → SSN, fd.4 stID → address }



Normal Forms - BCNF

- Theorem:
  given a schema R and a set of FD 'F', we can always decompose it to schemas R1, … Rn,
  so that
  - R1, …, Rn are in BCNF and
  - the decompositions is lossless
- But, some decompositions might lose dependencies ⇒ use 3NF
  - 3NF always loseless
  - 3NF always preserves dependencies

BCNF & Dependency Preservation

- BCNF is not always dependency preserving
- Example: 3NF but not BCNF

  PATIENT-VISIT (<u>PATIENT, HOSPITAL</u>,DOCTOR)

  fd.1 (Patient, Hospital) → Doctor

  fd.2 Doctor → Hospital
- Possible Decomposition 1:
  - Doctor-Hospital (Doctor, Hospital) {Doctor → Hospital}
  - Patient-Doctor (Patient, Doctor) {Patient → Doctor}
- Possible Decomposition 2:
  - Doctor-Hospital (Doctor, Hospital) {Doctor → Hospital}
  - Patient-Hospital (Patient, Hospital) {Patient → Hospital}
- BUT these decompositions lose fd.1

Normal Forms - 4NF

- **<u>Fourth Normal Form:</u> A relation is in BCNF and has no Multivalue Dependencies**
- Example: **FDC (<u>FACULTY, Dept, Committee</u>)**

1. A faculty member can belong to more than one dept.
2. A faculty can be on several **college-wide committees**.
3. There is **no relation** between dept. and committee.

FacultyID	Dept	Committee
F101	CS	Budget
F101	CoE	Budget
F101	CS	Curriculum
F101	CoE	Curriculum
F221	Bio	Library
F330	Math	Budget
F330	Math	Admissions

- Anomalies? Change F101 from Budget to Admissions

Normal Forms - 4NF

- Anomalies? Change F101 from Budget to Admissions

PK: FACULTY, Committee

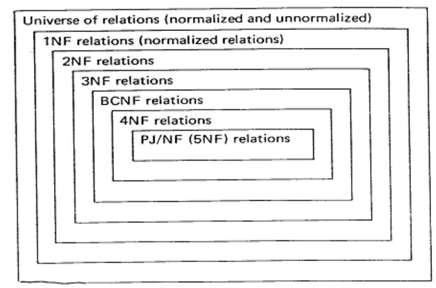
FacultyID	Committee
F101	Admissions
F101	Curriculum
F221	Library
F330	Budget
F330	Admissions

PK: FACULTY, Dept

FacultyID	Dept
F101	CS
F101	CoE
F221	Bio
F330	Math

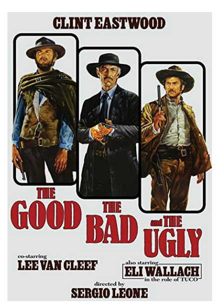
More Normal Forms…

- 5NF: Fifth Normal Form
  - No Join Dependencies
- 6NF: ….



Universal Relational Approach

- One single, large table
- Simple ?
- Good ? or Bad? Or just Ugly?
- Normalize it!



The Normalization Process

- The process of finding good (stable) set of relations that is a faithful model of the enterprise.
- **Decomposition (top-down process)**
  - start with a universal relation
  - identify functional dependencies
  - identify key(s)
  - If necessary, use decomposition to split the universal relation into a set of relations

Deriving the Keys of R from its FDs

- Let $X \subseteq R$, X^+ is its closure
- Algorithm for finding the keys
  - $X^+ = X$;
  - If $\exists$ an used $U \rightarrow V$ in FDs and $U \subseteq X^+$
    then goto step 3; else STOP;
  - $X^+ = X^+ \cup V$;
  - If $X^+ = R$
    then STOP; else goto step 2;
- A primary key is one of the minimal keys

Deriving the Keys of R from its FDs

- Example 1: $\boldsymbol{R} = (A, B, C)$, $\boldsymbol{F} = \{A \rightarrow B, B \rightarrow C\}$
- Derivation… all possible expansions & combinations

A^+: $A \rightarrow AB \rightarrow ABC$ ✓
B^+: $B \rightarrow BC \rightarrow$ ✘
C^+: $C \rightarrow$ ✘

AB^+: $AB \rightarrow ABC$ ✓
AC^+: $AC \rightarrow ABC$ ✓
BC^+: $BC \rightarrow$ ✘

ABC^+: $ABC \rightarrow ABC$ ✓

Deriving the Keys of R from its FDs

- Example 2: $\boldsymbol{R} = (A, B, C, D)$, $\boldsymbol{F} = \{A \rightarrow B, C \rightarrow D\}$
- Derivation… all possible expansions & combinations

A^+: $A \rightarrow AB \rightarrow$ ✘
B^+: $B \rightarrow$ ✘
C^+: $C \rightarrow CD \rightarrow$ ✘
D^+: $D \rightarrow$ ✘

AB^+: $AB \rightarrow AB \rightarrow$ ✘
AC^+: $AC \rightarrow ABC \rightarrow ABCD$ ✓
AD^+: $AD \rightarrow ABD \rightarrow$ ✘
BC^+: $BC \rightarrow BCD \rightarrow$ ✘
BD^+: $BD \rightarrow$ ✘
CD^+: $CD \rightarrow$ ✘

ABC^+: $ABC \rightarrow ABCD$ ✓
ABD^+: $ABD \rightarrow$ ✘
ACD^+: $ACD \rightarrow ABCD$ ✓
BCD^+: $BCD \rightarrow$ ✘

$ABCD^+$: $ABCD \rightarrow ABCD$ ✓

Optimization in Deriving the Keys from FDs

- Example 2: $\boldsymbol{R} = (A, B, C, D)$, $\boldsymbol{F} = \{A \rightarrow B, C \rightarrow D\}$
- Observation: If an attribute appears in RHS of a FD, it means that it cannot be part of the minimal key.
- Let $X = \cup_i RHS_i$, then Key = $R - X$
- Key = (A, B, C, D) – (B, D) = (A, C)

A^+: $A \rightarrow AB \rightarrow$ ✘
B^+: $B \rightarrow$ ✘
C^+: $C \rightarrow CD \rightarrow$ ✘
D^+: $D \rightarrow$ ✘

AB^+: $AB \rightarrow AB \rightarrow$ ✘
AC^+: $AC \rightarrow ABC \rightarrow ABCD$ ✓
AD^+: $AD \rightarrow ABD \rightarrow$ ✘
BC^+: $BC \rightarrow BCD \rightarrow$ ✘
BD^+: $BD \rightarrow$ ✘
CD^+: $CD \rightarrow$ ✘

BCNF Decomposition Algorithm

result := $\{R\}$;
done := false;
compute F^+;
while (not *done*) **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial FD that holds on R_i
      such that $\alpha \rightarrow R_i$ is not in F^+, and $\alpha \cap \beta = \emptyset$;
        $result := (result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$;
      end
    else *done* := **true;**

Note: each R_i is in BCNF, and decomposition is lossless-join.

Universal Relation: Example of BCNF Decomposition

- ***R*** = *(branch_name, branch_city, assets, customer, loan_number, amount)*
  F = *{branch_name* → *(assets, branch_city), loan_number* → *(amount, branch_name)}*
  Key = *{loan_number, customer}*
- Decomposition
  - R_1 = (<u>branch_name</u>, *assets, branch_city)*
  - R_2 = *(branch_name,* <u>customer, loan_number</u>, *amount)*
  - R_3 = *(branch_name,* <u>loan_number</u>, *amount)*
  - R_4 = (<u>customer, loan_number</u>)
- Final decomposition: R_1, R_3, R_4

Synthesis (bottom-up process)

- Begin with attributes and combine them into related group using functional dependencies to develop a set of normalized relations.
- A synthesis algorithm was developed by Bernstein.
- Basic steps:
  - make a list of all FDs
  - groups together those with the same determinant
  - construct a relation of each group.

Synthesis... Elaborated steps

1. Make a list of all FDs.
2. Eliminate extraneous attributes in each FD.
3. Remove any redundant FDs and find a non redundant covering of the input FDs.
   - Combine FD groups with equivalent key.
4. Group together those with the same determinant.
5. Construct a relation for each group.
6. If none of the resulting relations has the same key as one of the keys of the original universal relation, we add another relation containing one of the keys, with an empty FD set.

Example 1: Consider the following set of FDs.

f1: A ------> B
f2: A -------> C
f3: B -------> C
f4: B -------> D
f5: D -------> B
f6: ABC ------> F

Q: Using Synthesis construct a set 3NF relations.

Example 1 Solution

f1: A→B, f2: A→C } => A→BC union => f1: A→ABC, f6: ABC→F } A→F (f6')

f1: A→B, f3: B→C } => A→C trans Which is f2 so f2 is redundant

f1: A→B
f6': A→F
f3: B→C
f4: B→D
f5: D→B

R1 (A,B,F), R2 (B,C,D), R3 (D, B) } BCNF (3NF)

R2: Does not contain transitive FD

R1 (A,B,F), R4 (B,C,D) } BCNF (3NF)

R4: Contains transitive FD (f5)

Key = {A,B,C,D,F} - {B,F,C,D} = A

	A	B	C	D	F
R1	K	K	~~U~~ K	~~U~~ K	K
R4	U	K	K	K	U

Example 2: Consider the following set of FDs.

f1: A ------> BC
f2: A -------> D
f3: B -------> C
f4: C -------> D
f5: DE -------> C
f6: BC ------> D

Q: Using Synthesis construct a set 3NF relations.

Design Goals

- Goal:
  - BCNF
  - Lossless join
  - Dependency preservation
- If we cannot achieve this, we accept one of
  - Lack of dependency preservation
  - Redundancy due to use of 3NF
- Note, SQL does not provide a direct way of specifying FDs other than superkeys.
  - Can specify FDs using assertions, but they are expensive
  - Hard to test a FD whose left hand side is not a key

Conclusions

- Disadvantages of Normal Forms
  - It is not constructive. It does not provide a way to get a good design.
  - It can only be applied after we have a schema and tell if it is good or not.
  - It provides no conceptual design. It deals directly with relations and attributes.
  - It can be applied (more or less) only to relational schemas.
- Schema Design: Conceptual Design
  - Entity-Relationship (E-R) or UML
- In practice, E-R diagrams usually lead to tables in BCNF