Objectives

- The purpose of normalization
- Data redundancy and Update Anomalies
- Functional Dependencies
- The Process of Normalization
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)

Objectives (continue ...)

- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4NF)
- Fifth Normal Form (5NF)

Review: Database Design

- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - high level descr (often done w/ER model)
- Logical Design
  - translate ER into DBMS data model
- Schema Refinement
  - consistency,normalization
- Physical Design - indexes, disk layout
- Security Design - who accesses what

The Purpose of Normalization

Database normalization is the process of removing redundant data from your tables in to improve storage efficiency, data integrity, and scalability

The process of normalization is a formal method that identifies relations based on their primary or candidate keys and the functional dependencies among their attributes.

Update Anomalies

Relations that have redundant data may have problems called **update anomalies**, which are classified as ,

Insertion anomalies
Deletion anomalies
Modification anomalies

Example of Update Anomalies

To insert a new staff with branchNo B007 into the StaffBranch relation;

To delete a tuple that represents the last member of staff located at a branch B007;

To change the address of branch B003.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St,Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St,Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St,Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

LHI Aman

Example of Update Anomalies

Staff

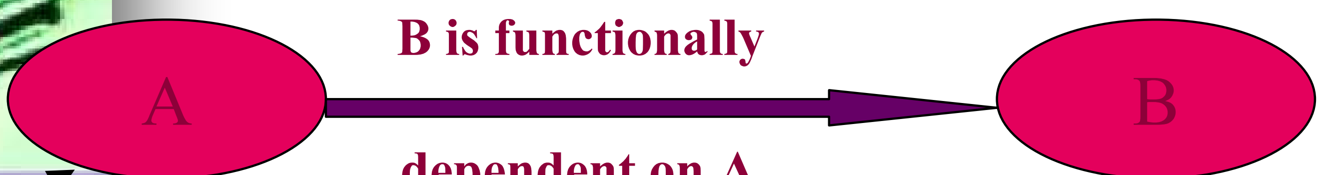
staffNo	sName	position	salary	branceNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branceNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St,Glasgow

Functional Dependencies

Functional dependency describes the relationship Between attributes in a relation.
For example, if A and B are attributes of relation R, and B is functionally dependent on A (denoted A$\longrightarrow$B), if each value of A is associated with exactly one value of B. (A and B may each consist of one or more attribute



Determinant

Refers to the attribute or group of attributes on the left-hand side of the arrow of a functional dependency

Functional Dependencies

Trival functional dependency means that the right-hand side is a subset (not necessarily a proper subset) of the left-hand side.

For example:

staffNo, sName → sName
staffNo, sName → staffNo

They do not provide any additional information about possible integrity constraints on the values held by these attributes.

We are normally more interested in **nontrivial dependencies** because they represent integrity constraints for the relation.

Functional Dependencies

Main characteristics of functional dependencies in normalization :

- Have a one-to-one relationship between attribute(s) on the left- and right- hand side of a dependency;
- hold for all time;
- are completely nontrivial.

Specifying FD’s for a Relation

- Amstrong’s Interference Rules
- Closure set of Attributes
- Closure set of FD’s

Amstrong's Interference Rules

- Armstrong's Axioms (X, Y, Z are <u>sets</u> of attributes):
  - *<u>Reflexivity</u>*: If $X \supseteq Y$, then $X \rightarrow Y$
  - *<u>Augmentation</u>*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - *<u>Transitivity</u>*: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- These are *sound* and *complete* inference rules for FDs!
  - i.e., using AA you can compute all the FDs in F+ and only these FDs.
- Some additional rules (that follow from AA):
  - *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - *Decomposition*: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
  - *Composition : If* $X \rightarrow Y$, and $Z \rightarrow A$, Then $XZ \rightarrow YA$
  - **Self-determination: X** $\rightarrow X$

Closure Set of Attributes

- Computing the closure of a set of FDs can be expensive.
- Typically, we just want to check if a given FD $X \rightarrow Y$ is in the closure of a set of FDs F. An efficient check:
  - Compute [attribute closure]() of X (denoted X^+) wrt F. $X^+ =$ Set of all attributes A such that $X \rightarrow A$ is in F^+
    - $X^+ := X$
    - Repeat until no change: if there is an fd $U \rightarrow V$ in F such that U is in X^+, then add V to X^+
  - Check if Y is in X^+
  - Approach can also be used to find the keys of a relation.
    - If all attributes of R are in the closure of X then X is a superkey for R.
    - Q: How to check if X is a "candidate key"?

Algorithm to compute Closure

- Let X be a set of attributes that will become closure
- Now repeatedly search for all FD's of the form X -> Y such that

  If X is a part of closure but Y is not a part of closure, then add Y to the closure
- Repeat above step, as many times as necessary untill no more attributes can be added
- Then, set X will become a closure set of attributes

Example :

- Given FD's are
- A -> B
  B -> C
  C -> E
  F -> G

  Compute $\mathbf{A}^+$

  X = A
  = AB
  = ABC
  = ABCE

A^+ = ABCE , A -> ABCE

Applications of Closure of Attributes

- It is used to identify additonal FD’s
- It is used to identify candidate keys for the relation
  - If any closure cover all the attributes, it acts as a key
- It is used to find out equivalences of FD’s
- It is used to identify Irreducible set of FD’s or canonical form of FD’s

Functional Dependencies

Identifying the primary key

Functional dependency is a property of the meaning or semantics of the attributes in a relation. When a functiona dependency is present, the dependency is specified as a **constraint** between the attributes.

An important integrity constraint to consider first is **the identification of candidate keys, one of which is selected to be the primary key** for the relation using functional dependency.

Attribute Closure (example)

- $R = \{A, B, C, D, E\}$
- $F = \{ B \rightarrow CD, D \rightarrow E, B \rightarrow A, E \rightarrow C, AD \rightarrow B$
- Is $B \rightarrow E$ in F^+ ?

  $\mathbf{B^+ = B}$

  $B^+ = BCD$

  $B^+ = BCDA$

  $B^+ = BCDAE$ …

  and B is a key for R too!

  $A^+ = A$

  $C^+ = C$
- Is D a key for R?

  $D^+ = D$

  $D^+ = DE$

  $D^+ = DEC$

  … No

- **Is AD a key for R?**

  $AD^+ = AD$

  $AD^+ = ABD$ and B is a key
- **Is AD a *candidate* key for R?**

  $A^+ = A$, $D+ = DEC$

  … A,D not keys
- **Is ADE a *candidate* key for R?**

  … No! AD is a key, so ADE i

  superkey, but not a cand. ke

  $ADE^+ = ADECB$

Equivalences of FD’s

- Consider two set of FD’s like

  F1 and F2

  They are considered as Equivalent if

  Closure of F1 = Closure of F2

i.e, Every FD’s in F1 can be derived from F2 and Every FD’s in F2 can be Derived from F1

Example

F

A->C
AC->D
E->AD
E->H

G

A->CD
E->AH

- Steps:
  - Take G set and compute Closure from F.Test whether they can be implied
  - Take F set and compute Closure from G.Test whether they can be implied

Example

- G :

  $A^+ = ACD$ (A -> CD)

  $E^+ = EADH$ (E -> AH)

All the FD's of G can be derived from F.

- F :

  $A^+ = ACD$ (A -> C)

  $AC^+ = ACD$ (AC -> D)

  $E^+ = EACDH$ (E->AD, E->H)

All the FD's of F can be derived from G.

F is Equivalent to G

Minial Sets of Functional Dependencies

A set of functional dependencies X is **minimal** if it satisfies the following condition:

- Every dependency in X has a single attribute on its right-hand side

- We cannot replace any dependency A → B in X with dependency C →B, where C is a proper subset of A, and still have a set of dependencies that is equivalent to X.

- We cannot remove any dependency from X and still have a set of dependencies that is equivalent to X.

Steps to Identify the Irreducible set

1. Try to have single attribute on RHS.

2. Remove the inessential FD's like X -> A, Then Compute X^+ from the remaining FD's and check whether this X^+ includes A. If it is true, then removal of X -> A will not have any influence in the database design.

3. If LHS consists multiple attributes like

   XY -> Z, remove X and compute Y^+ ,or remove Y and compute X^+ from the available FD's. If they have equal attributes, then removal of these attributes will not have any influence.

4. Apply Union rule to the common LHS attribute

Example :

A -> B
C -> B
D -> ABC
AC -> D

Step 1 :

A -> B ------------- 1

C -> B ------------- 2

D -> A ------------- 3

D -> B ------------- 4

D -> C ------------- 5

AC -> D -----------6

Step 2 :

- Remove 1 { compute A^+ from 2,3,4,5,6 }

  A^+ = A ------ Can't remove 1

  Remove 2 {compute C^+ from 1,3,4,5,6 }

  C^+ = C ------ Can't remove 2

  Remove 3 {compute D^+ from 1,2,4,5,6 }

  D^+ = DBC ------ Can't remove 3

  Remove 4 {compute D^+ from 1,2,3,5,6 }

  D^+ = DABC ---- Can remove 4

  Remove 5 {compute D^+ from 1,2,3,4,6}

  D^+ = DAB ------ Can't remove 5

Step 3 :

A -> B C -> B D -> A D -> C AC -> D

C^+ = CB, A^+ = AB,

- Remove A from 6, Compute C^+ .

  $C^+ = CDAB$

- Remove C from 6, Compute A^+ .

  $A^+ = CDAB$

Can't remove A or C.

Canonical Form of FD

A -> B
C -> B
D -> A
D -> C
AC -> D

Example of A Minial Sets of Functional Dependencies

A set of functional dependencies for the StaffBranch Relation satisfies the three conditions for producing a minimal set.

staffNo → sName
staffNo → position
staffNo → salary
staffNo → branchNo
staffNo → bAddress
branchNo → bAddress
branchNo, position → salary
bAddress, position → salary

Types Of Functional Dependencies

- Partial
- Transitive
- Full

Partial Functional Dependency

- One or two non-key attributes functionally dependent on a part of Primary key.
- Whenever there is a P.D, it causes redundancy and retrival problem.

Ex: AB -> C   R=ABCD,

B -> D

AB^+ = ABCD ----> AB is the Key

B -> D : D (Non key) depends on B(part of key)

Hence , P.D

Conditions for No P.D :

Under the Following Conditions, there is no P.D :

- If P.key consists of only one attribute.
- If table consists of only two attribute
- If all the attributes of the table forms the P.key of the table

Transitive Functional Dependency

- If one non-key attribute determines another non-key attributes, then T.D
- Each attributes must depend directly on the primary key; all attributes that are not dependant upon the primary key must be eliminated
- Ex: AB -> C R=ABCDE
- B -> D PD
- C -> E TD

A -> B
B -> C

AB^+ = ABCDE ---- AB is the Key

C,D,E ---- Non key attributes

Hence , T.D

Conditions for No T.D :

Under the Following Conditions, there is no T.D :

- If all the attributes in the Relation are part of the P.Key.
- If the Relation consists of only two attributes

Full functional dependency

Full functional dependency indicates that if A and B are attributes of a relation, B is fully functionally dependent on A if B is functionally dependent on A, but not on any proper subset of A.

A functional dependency A→B is **partially dependent** if there is some attributes that can be removed from A and the dependency still holds.

The Process of Normalization

- Normalization is often executed as a series of steps. Each step corresponds to a specific normal form that has known properties.

- As normalization proceeds, the relations become progressively more restricted in format, and also less vulnerable to update anomalies.

- For the relational data model, it is important to recognize that it is only first normal form (1NF) that is critical in creating relations.
  All the subsequent normal forms are optional.

First Normal Form (1NF)

Repeating group = (propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

ting groups.

ClientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John kay	PG4 PG16	6 lawrence St,Glasgow 5 Novar Dr, Glasgow	1-Jul-00 1-Sep-02	31-Aug-01 1-Sep-02	350 450	CO40 CO93	Tina Murphy Tony Shaw
CR56	Aline Stewart	PG4 PG36 PG16	6 lawrence St,Glasgow 2 Manor Rd, Glasgow 5 Novar Dr, Glasgow	1-Sep-99 10-Oct-00 1-Nov-02	10-Jun-00 1-Dec-01 1-Aug-03	350 370 450	CO40 CO93 CO93	Tina Murphy Tony Shaw Tony Shaw

Definition of First Normal Form (1NF)

1. The cells of the table must have a single value.
2. Neither repeating groups nor arrays are allowed as values.
3. All entries in any column must be of the same kind.
4. Each column must have a unique name
5. No two rows in a table are identical

1NF (Continue ...)

First Normal Form is a relation in which the intersection of each row and column contains one and only one value.

There are two approaches to removing repeating groups from unnormalized tables:

1. Removes the repeating groups by entering appropriate data in the empty columns of rows containing the repeating data.

2. Removes the repeating group by placing the repeating data, along with a copy of the original key attribute(s), in a separate relation.
   A primary key is identified for the new relation.

1NF ClientRental relation with the first approach

With the first approach, we remove the repeating group (property rented details) by entering the appropriate clien data into each row.

The ClientRental relation is defined as follows,

ClientRental (<u>clientNo</u>, <u>propertyNo</u>, cName, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

ClientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

1NF ClientRental relation with the second approach

With the second approach, we remove the repeating group (property rented details) by placing the repeating data along with a copy of the original key attribute (clientNo) in a separte relation.

Client (clientNo, cName)

PropertyRentalOwner (clientNo, propertyNo, pAddress, rentStart, rentFinish, rent, ownerNo, oName)

ClientNo	cName
CR76	John Kay
CR56	Aline Stewart

ClientNo	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	6 lawrence St,Glasgow	1-Jul-00	31-Aug-01	350	CO40	Tina Murphy
CR76	PG16	5 Novar Dr, Glasgow	1-Sep-02	1-Sep-02	450	CO93	Tony Shaw
CR56	PG4	6 lawrence St,Glasgow	1-Sep-99	10-Jun-00	350	CO40	Tina Murphy
CR56	PG36	2 Manor Rd, Glasgow	10-Oct-00	1-Dec-01	370	CO93	Tony Shaw
CR56	PG16	5 Novar Dr, Glasgow	1-Nov-02	1-Aug-03	450	CO93	Tony Shaw

Definition of Second Normal Form (2NF) : Partial Dependencies

- Anomalies can occur when attributes are dependent on only part of a multi-attribute key
- A relation is in 2NF, when all Non-Key attributes are dependent on the whole key.
- No attribute is dependent on only a part of key
- Any relation having a single attribute is in 2NF

Second Normal Form (2NF)

Second normal form (2NF) is a relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on the primary key.

The normalization of 1NF relations to 2NF involves The removal of partial dependencies. If a partial dependency exists, we remove the function dependent attributes from the relation by placing them in a new relation along with a copy of their determinant.

2NF ClientRental relation

The ClientRental relation has the following functional dependencies:

fd1 : clientNo, propertyNo → rentStart, rentFinish (Primary Key)

fd2 : clientNo → cName (Partial dependency)

Fd3: propertyNo → pAddress, rent, ownerNo, oName (Partial dependency)

Fd4: ownerNo → oName (Transitive Dependency)

Fd5: clientNo, rentStart → propertyNo, pAddress,rentFinish, rent, ownerNo, oName (Candidate key)

Fd6: propertyNo, rentStart → clientNo, cName, rentFinish (Candidate key)

2NF ClientRental relation

After removing the partial dependencies, the creation of the three new relations called Client, Rental, and PropertyOwner

Client (clientNo, cName)

Rental (clientNo, propertyNo, rentStart, rentFinish)

PropertyOwner (propertyNo, pAddress, rent, ownerNo, oName)

Client

ClientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

ClientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-00	31-Aug-01
CR76	PG16	1-Sep-02	1-Sep-02
CR56	PG4	1-Sep-99	10-Jun-00
CR56	PG36	10-Oct-00	1-Dec-01
CR56	PG16	1-Nov-02	1-Aug-03

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 lawrence St,Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	370	CO93	Tony Shaw

Figure 6 2NF ClientRental relation

Definition of Third Normal Form (3NF) : Transitive Dependencies

- Anomalies can occur when a relation contains one or more transitive dependencies
- A transitive dependency exists when A->B. and A is not a part of Key.

Third Normal Form (3NF)

A relation that is in 1NF and 2NF, and in which no non-primary-key attribute is **transitively** dependent on the primary key.

The normalization of 2NF relations to 3NF involves the removal of transitive dependencies by placing the attribute(s) in a new relation along with a copy of the determinant.

Boyce-Codd Normal Form (BCNF)

A relation is in BCNF, if and only if, every determinant is a candidate key.

The difference between 3NF and BCNF is that for a Functional dependency A → B, 3NF allows this dependency in a relation
if B is a primary-key attribute and A is not a candidate key,
whereas BCNF insists that for this dependency to remain in a relation, A must be a candidate key.

Examples:

- R=ABCDEFGHIJ

AB -> C
A -> DE
B -> F
F -> GH
D -> IJ

Examples:

- R=ABCDEFGHIJ

AB -> C
A -> DE -------------- P.D
B -> F ----------------- P.D
F -> GH -------------- T.D
D -> IJ ---------------- T.D

AB ----- > Key

Result : upto 3NF

ADE
BF
ABC
DIJ
FGH

Examples:

- R=ABCDEFGH

AB -> CEFGH
A -> D P.D
F->G T.D
FB->H
HBC->ADEFG
FBC->ADE

Upto 2NF

- AB = P.Key

AD
ABCEFGH

Upto 3NF

AD
FG
ABCEFH

Upto BCNF

AD
FG
FBH
ABCEF

Examples:

- R=ABCDEFG

BCD -> A
BC -> E
A -> F
F -> G
C -> D
A -> G

Examples:

- R=ABCEDFGH

A -> BC
ABE -> CDGH
C -> GD
D -> G
E -> F

Examples:

- R=ABCEDFGH

A -> BC
ABE -> CDGH
C -> GD
D -> G
E -> F

Examples:

- R=ABCEDFGH

A -> BC
ABE -> CDGH
C -> GD
D -> G
E -> F

Fourth Normal Form (4NF)

Multi-valued dependency (MVD)

represents a dependency between attributes (for example, A, B and C) in a relation, such that for each value of A there is a set of values for B and a set of value for C. However, the set of values for B and C are independent of each other.

A multi-valued dependency can be further defined as being trivial or nontrivial.

A MVD A →> B in relation R is defined as being trivial if

- B is a subset of A

or

- A U B = R

A MVD is defined as being nontrivial if neither of the Two conditions is satisfied.

Fourth Normal Form (4NF)

A relation that is in Boyce-Codd normal form and Contains no nontrivial multi-valued dependencies.

Example : Emp

Emp_Name	Emp_Skills	Emp_language
Hari	C	Punjabi
Shyam	Java	Hindi
Ram	J2EE	Gujrati
Rahul	J2ME	Tamil

Upto 4NF :

Emp_Skills

Emp_ Name	Emp_ Skills
Hari	C
Shyam	java
Ram	J2EE
Rahul	J2ME

Emp_Languages

Emp_ Name	Emp_ Language
Hari	Punjabi
Shyam	Hindi
Ram	Gujrati
Rahul	Tamil

Lossless-join dependency

- A property of decomposition, which ensures that
- no spurious tuples are generated when relations are reunited through a natural join operation.

Example :

R

A	B	C
a1	b1	c1
a2	b2	c2
a3	b1	c3

Example :

R

<u>A</u>	B	C
a1	b1	c1
a2	b2	c2
a1	b3	c3

R1

A	B
a1	b1
a2	b2
a1	b3

R2

A	C
a1	c1
a2	c2
a1	c3

- R1 × R2 :

A	B	C
a1	b1	c1
a2	b2	c2
a1	b3	c3
a1	b1	c3
a1	b3	c1

- R1 × R2 :

A	B	C
a1	b1	c1
a2	b2	c2
a1	b3	c3
a1	b1	c3
a1	b3	c1

- R1 × R2 :

A	B	C
a1	b1	c1
a2	b2	c2
a1	b3	c3
a1	b1	c3
a1	b3	c1

R

Additional Tuples

- Hence , Lossy Decomposition
- Ex : R (ABC),
  R1 (AB) , R2(AC)
  B -> C

R1 Λ R2 = A
Check
R1 Λ R2 -> R1 : A -> AB
or
R1 Λ R2 -> R2 : A -> AC

If true, Lossless Decompsition,
Else Lossy Decompsition

Fifth Normal Form (5NF)

A relation that has no join dependency.

Join dependency

Describes a type of dependency. For example, for a relation R with subsets of the attributes of R denoted as A, B, …, Z, a relation R satisfies a join dependency if, and only if, every legal value of R is equal to the join of its projections on A, B, …, Z.

Example :

Ambassador	Company	Product
Amir	Coca Cola	Coke
Kareena	Boroplus	Cold Cream
Katrina	Coca Cola	Maaza
Katrina	Coca Cola	Coke
Kareena	Boroplus	Body Lotion

FIFTH NORMAL FORM (5 NF)

Ambassador	Company
Amir	Coca Cola
Kareena	Boroplus
Katrina	Coca Cola

Company	Product
Coca Cola	Coke
Coca Cola	Maaza
Boroplus	Cold Cream
Boroplus	Body lotion

Ambassador	Product
Amir	Coke
Kareena	Cold Cream
Kareena	Body lotion
Katrina	Coke
Katrina	Maaza

Advantages of Normalization

- It reduces the redundancy within a table
- Increases Consistency

Disadvantages of Normalization

- It can not detect redundancy between the relations
- The Fragmented relation from the normalization process may not have real world Meaning
- It slows down the query retrival process because it requires Join and subqueries

THE END