Chapter – 7
Relational Database Design

- First Normal Form
- Pitfalls in Relational Database Design
- Functional Dependency
- Normalization

First Normal Form

- Domain is atomic if its elements are considered to be indivisible units
  - Example of atomic domain

    set of integers forms atomic domain
  - Examples of non-atomic domains:
    - Set of names, composite attributes like address
- A relational schema R is in first normal form (1NF) if the domains of all attributes of R are atomic
- If relation schemas had an attribute whose domain is non atomic,schema would not be in first normal form

- Suppose that employee in organization are given emp id as *CS0012* or *EE1127*
- If the first two characters are extracted to find the department, the number indicates number of employee within department.
- If these are used as primary key and suppose employee change department, his id must be changed every where it occurs.

- The requirements to satisfy the 1st NF:
  - values in each column of a table are atomic (No multi-value attributes allowed).
  - There are no repeating groups: two columns do not store similar information in the same table.
  - E.g 1 NF

branch-name	*branch-city*	*assets*	*customer-name*	*loan-number*	*amount*
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

Pitfalls in Relational Database Design

- Relational database design requires that we find a "good" collection of relation schemas. A bad design may lead to
  - Repetition of Information.
  - Inability to represent certain information

Example

- Consider the relation schema:

  Lending-schema = (*branch-name, branch-city, assets,*
  customer-name, loan-number, amount)

branch-name	*branch-city*	*assets*	*customer-name*	*loan-number*	*amount*
Downtown	Brooklyn	9000000	Jones	L-17	1000
Redwood	Palo Alto	2100000	Smith	L-23	2000
Perryridge	Horseneck	1700000	Hayes	L-15	1500
Downtown	Brooklyn	9000000	Jackson	L-14	1500

- Redundancy:
  - Data for *branch-name, branch-city,* assets are repeated for each loan that a branch makes
  - Wastes space
  - If we add new record (Perryridge, Horseneck, 1700000, Adams, L-31, 1500) so branch name horseneck, 1700000 is repeated.
  - Complicates updating, introducing possibility of inconsistency of *assets* value
  - For e.g , that the assets of the Perryridge branch change from 1700000 to 1900000.

- So in upatation of above table [without normalize]
- it must ensure that every tuple pertaining to the Perryridge branch is updated, or else our
- database will show two different asset values for the Perryridge branch. Which may lead database in inconsistent state.
- functional dependency
- branch-name → assets that is assests value is depends upon the branch_name

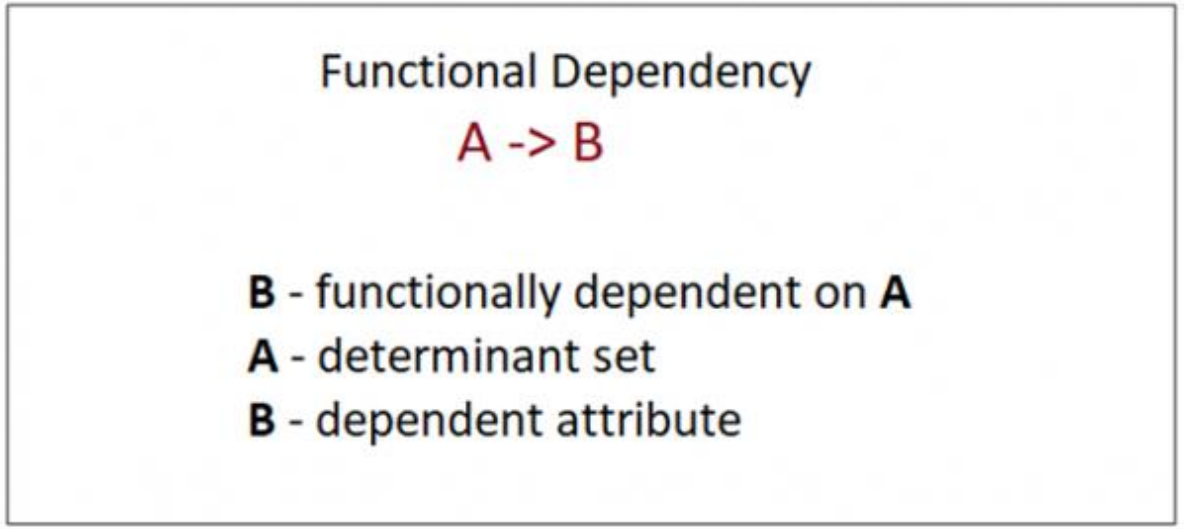
- Null values
  - Cannot store information about a branch if no loans exist
  - L-25 loan is available in ‘xyz’ branch, city newyork assest is 5000000
  - If no customer for L-25 loan , record for branch xyz will not be inserted into table.

Functional Dependencies

- It helps in distinguishing between good database design and bad database design
- Constraints on the set of legal relations.
- It Require that the value for a certain set of attributes determines uniquely the value for another set of attributes.
- A functional dependency is a generalization of the notion of a *key*.
- Functional Dependency (FD) is a constraint that determines the relation of one attribute to another attribute in a Database Management System (DBMS).
- Functional Dependency helps to maintain the quality of data in the database

- A functional dependency is denoted by an arrow "→".

- 1
- Deptid name
- 1 x
- 2 y
- 3 z



- for same value A , value of B should be same.
- for different A, value of B could be same or different

- Example , deptid , deptName . deptId is primary key.
- So from each deptid can uniquely identifies the DeptName attribute(and whole tuple)
- Deptid->DeptName

- So
- DeptId -> DeptName
- deptName is functionally dependent on DeptID

- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:

- Loan-info-schema = (loan-number, branch-name, customer-name, amount,assets)
- dependencies that we expect to hold on this relation schema is
  - loan-number → amount
  - loan-number → branch-name
  - branch-name → assest

- would not expect the following to hold
  - loan-number → customer-name
- Same loan can be taken from more than one customer. (from loan number we can not uniquely identify the customer name.

- E.g Let us consider the relation r
- . Observe that A → C is satisfied.
- There are two tuples that have an
- A value of a1.
- These tuples have the same C value
- c1.

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_2	c_2	d_3
a_3	b_3	c_2	d_4

- Similarly, the two tuples with an A value of a2 have the same C value, c2
- The **functional dependency C → A is not satisfied,**
- Check tuple s t1 = (a2, b2, c2, d3) and t2 = (a3, b3, c2, d4).
- These two tuples have the same C values, c2, but they have different A values, a2 and a3, respectively.
- Thus, we have found a pair of tuples t1 and t2 such that t1[C] = t2[C], but t1[A] ≠ t2[A]. So c → A [a is not depends on value of c]
- s t1 = (a2, b2, c2, d2) and t2 = (a2, b2 c2, d3). A ->C

- that t1[A] = t2[A], then t1[C] = t2[C].

Functional dependency of bank db given into book

- • On Branch-schema:
  - branch-name → branch-city
  - branch-name → assets
- • On Customer-schema:
  - customer-name → customer-city
  - customer-name → customer-street
- • On Loan-schema:
  - loan-number → amount
  - loan-number → branch-name
- • On Borrower-schema:
  - No functional dependencies
- • On Account-schema:
  - account-number → branch-name
  - account-number → balance
- • On Depositor-schema:
  - No functional dependencies

- E.g of functional dependency

A -> A , A->B : for one value of a we can fine exactly one value of B

B->A : for single value of B there is no one value of A

So B->A is not possible. [value 2 for B there is more

Than one value of A [1,2,3] available]

T1[[b]=t2[b] but t1[a] not equalto t2[a]

A	B
1	2
2	2
3	2
4	3
5	2
6	3

Types of Functional Dependency used in Normalization

- Reflexive Axiom:
  - Always the trivial dependencies like X->X, Y->Y shall be true and valid for each and every instance of the relation.
- Decomposition Axiom
  - if there exist functional dependency X->YZ, then as per decomposition axiom, we can conclude => X->Y and X->Z are valid.
- Full dependency :
  - A dependency is said to be full if and only if the determinant of the dependency is either a candidate key or a super key.
  - X -> Y
  - X is either candidate key or super key
  - Y could be any attribute Prime or Non Prime.

- Full functional dependency is defined as all the non candidate key attributes fully depends upon primary key attribute.
- E.g Employee(id,name,city)

- Id is primary key , single value for name and city value can be get form id.
- Id->name , id->city, id->namecity
- Name is fully depends on id

id	name	City
1	x	A
2	x	B
3	y	A

- Partial Dependency
  - Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key.
  - R{ABCD},
  - functional dependencies AB->CD and A->C. AB is candidate key cd is non prime attribute.
  - [non prime attribute are attribute which uniquely can not identify any record , it is not part of candidate key]
  - AB determines the value of c and d AB->CD
  - A->C is A can determine value of C . A is subset of AB so
  - A->C is partial dependency. C

- X->Y where X is not the minimal set of attributes that uniquely determines Y. Some attributes could be removed from X, and the dependency would still hold.

- Transitive Dependency A->B B->C A->C
  - A type of functional dependency where an attribute is functionally dependent on an attribute other than the primary key.
  - Thus its value is only indirectly determined by the primary key.
  - A condition where A, B, and C are attributes of a relation such that
  - if A -> B and B ->C, then C is transitively dependent on A via B

- E.g
- A B
- MovieId->Movietype_ID
- B C
- Moivetype_ID->Moivetype_name
- A->B , B->C
- Moivetype_name is depends on
- Movietype_ID
- and
- Movietype_ID is depends on MovieID so

- MovieID-> Moivetype_name is transitive dependency

Movie_ID	Movietype_ID	Movietype_name	DVD_Price ($)
M08	L09	Crime	180
M03	L05	Drama	250
M05	L09	Crime	180

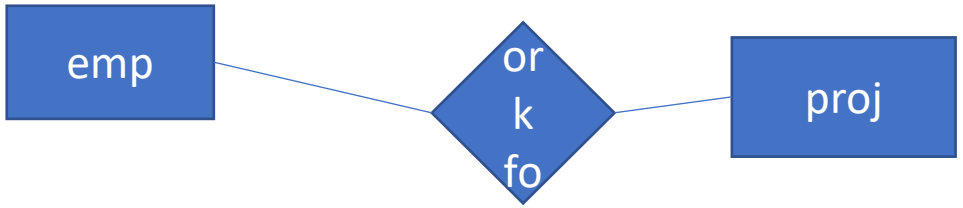
Normalization

- Normalization is the process of minimizing redundancy from a relation or set of relations. Redundancy in relation may cause insertion, deletion and updation anomalies.
- Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition)
- It is multi step process.

Rollno	name	Branch	hod	office_tel
401	Akon	CSE	Mr. X	53336
402	Bkon	CSE	Mr. X	53336
403	Ckon	CSE	Mr. X	53336
404	Dkon	CSE	Mr. X	53336
405	john	Physics	Mr.Y	56789

Modification Anomalies

- Deletion anomaly
  - deleting one fact about an entity deletes a fact about another entity
  - E.g if information of student 405 is deleted information of physics branch is also lost
- Insertion anomaly
  - cannot insert one fact about an entity unless a fact about another entity is also added
  - New branch information can not be added until some student not opt that branch
- Update anomaly
  - changing one fact about an entity requires multiple changes to a table
  - E.g changing a tel of CSE branch its needs to change in all 4 records



Scenario

A few employees works for one project.

Same employee can work in many project

Project Num : 18

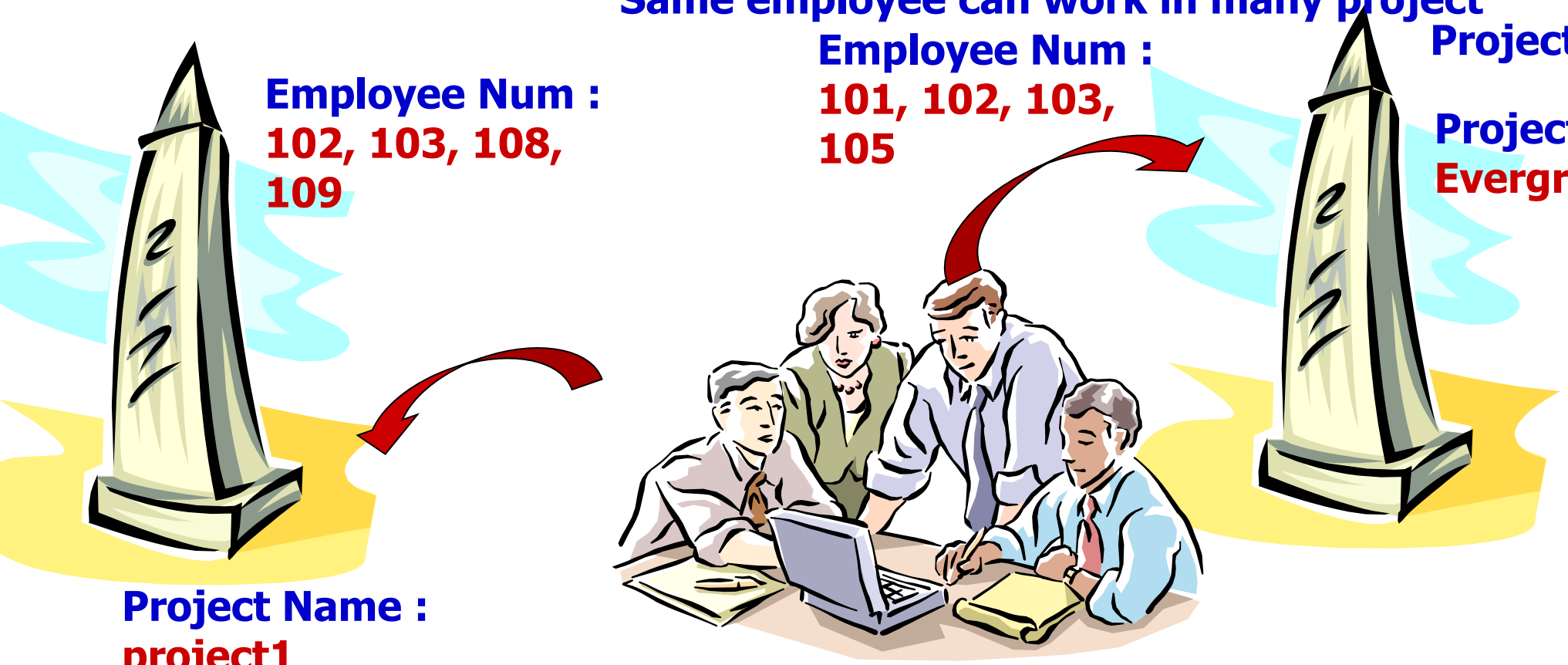
Employee Num : 102, 103, 108, 109

Project Name : project1

Employee Num : 101, 102, 103, 105

Project Num : 15

Project Name : Evergreen



Sample Form

Project Num : 15

Project Name : Evergreen

Emp Num	Emp Name	Job Class	Chr Hours	Hrs Billed	Total
101					
102					
103					
105					

TABLE 5.1 A SAMPLE REPORT LAYOUT

PROJ. NUM.	PROJECT NAME	EMPLOYEE NUMBER	EMPLOYEE NAME	JOB CLASS.	CHG/ HOUR	HOURS BILLED	TOTAL CHARGE
15	Evergreen	103	June E. Arbough	Elec. Engineer	$84.50	23.8	$2,011.10
		101	John G. News	Database Designer	$105.00	19.4	$2,037.00
		105	Alice K. Johnson *	Database Designer	$105.00	35.7	$3,748.50
		106	William Smithfield	Programmer	$35.75	12.6	$450.45
		102	David H. Senior	Systems Analyst	$96.75	23.8	$2,302.65
				Subtotal			**$10,549.70**
18	Amber Wave	114	Annelise Jones	Applications Designer	$48.10	24.6	$1,183.26
		118	James J. Frommer	General Support	$18.36	45.3	$831.71
		104	Anne K. Ramoras *	Systems Analyst	$96.75	32.4	$3,134.70
		112	Darlene M. Smithson	DSS Analyst	$45.95	44.0	$2,021.80
				Subtotal			**$7,171.47**
22		105	Alice K. Johnson	Database Designer	$105.00	64.7	$6,793.50
		104	Anne K. Ramoras	Systems Analyst	$96.75	48.4	$4,682.70
		113	Delbert K. Joenbrood*	Applications Designer	$48.10	23.6	$1,135.16
		111	Geoff B. Wabash	Clerical Support	$26.87	22.0	$591.14
		106	William Smithfield	Programmer	$35.75	12.8	$457.60
				Subtotal			**$13,660.10**
25		107	Maria D. Alonzo	Programmer	$35.75	24.6	$879.45
		115	Travis B. Bawangi	Systems Analyst	$96.75	45.8	$4,431.15
		101	John G. News *	Database Designer	$105.00	56.3	$5,911.50
		114	Annelise Jones	Applications Designer	$48.10	33.1	$1,592.11
		108	Ralph B. Washington	Systems Analyst	$96.75	23.6	$2,283.30
		118	James J. Frommer	General Support	$18.36	30.5	$559.98
		112	Darlene M. Smithson	DSS Analyst	$45.95	41.4	$1,902.33
				Subtotal			**$17,559.82**

First Normal Form (1NF)

- For a table to be in the First Normal Form, it should follow the following
  1. It should only have single(atomic) valued attributes/columns.
  2. Values stored in a column should be of the same domain (No multi-value attributes allowed).
  3. All the columns in a table should have unique names.
  4. There are no repeating groups: two columns do not store similar information in the same table.

- The Each table has a primary key: minimal set of attributes which can uniquely identify a record
- values in each column of a table are atomic (No multi-value attributes allowed).
- There are no repeating groups: two columns do not store similar information in the same table.

- Unnormalized table is converted into 1NF

- Project_employee(proj_no,proj_name,emp_no,emp_name,job_class,chr_hour,hour_work)

	PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CLASS	CHG_HOUR	HOURS
▶	15	Evergreen	103	June E. Arbough	Elect. Engineer	$84.50	23.8
	15	Evergreen	101	John G. News	Database Designer	$105.00	19.4
	15	Evergreen	105	Alice K. Johnson *	Database Designer	$105.00	35.7
	15	Evergreen	106	William Smithfield	Programmer	$35.75	12.5
	15	Evergreen	102	David H. Senior	Systems Analyst	$96.75	23.9
	18	Amber Wave	114	Annelise Jones	Applications Designer	$48.10	24.6
	18	Amber Wave	118	James J. Frommer	General Support	$18.36	45.3
	18	Amber Wave	104	Anne K. Ramoras *	Systems Analyst	$96.75	32.1
	18	Amber Wave	112	Darlene M. Smithson	DSS Analyst	$45.95	44.0
	22	Rolling Tide	105	Alice K. Johnson	Database Designer	$105.00	64.7
	22	Rolling Tide	104	Anne K. Ramoras	Systems Analyst	$96.75	48.9
	22	Rolling Tide	113	Delbert K. Joenbrood *	Applications Designer	$48.10	23.6
	22	Rolling Tide	111	Geoff B. Wabash	Clerical Support	$26.87	22.5
	22	Rolling Tide	106	William Smithfield	Programmer	$35.75	12.1
	25	Starflight	107	Maria D. Alonzo	Programmer	$35.75	24.7
	25	Starflight	115	Travis B. Bawangi	Systems Analyst	$96.75	45.8
	25	Starflight	101	John G. News *	Database Designer	$105.00	56.3
	25	Starflight	114	Annelise Jones	Applications Designer	$48.10	33.1

Functional Dependencies

1. If one set of attributes in a table determines another set of attributes in the table, then the second set of attributes is said to be functionally dependent on the first set of attributes.

Proj_NUM → proj_name
EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR
PROJ_NUM,EMP_NUM → HOURS

{

2nd Normal Form (2NF)

- Rules for table /database in 2 NF.

  1 . The table should be in the First Normal Form.

  2 . There should be no Partial Dependency.(remove partial dependency)

  Find the dependency first for given table

  Proj_NUM → PROJ_NAME -> PRATIAL DEPENDENCY

  EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR

  PROJ_NUM,EMP_NUM → HOURS

  So we can say

  Candidate keys set {PROJ_NUM,EMP_NUM} which uniquely identify the each record. So {PROJ_NUM,EMP_NUM }is candidate keys.

- And non prime attributes are → PROJ_NAME, EMP_NAME, JOB_CLASS, CHG_HOUR,Hour
- according to partial dependency definition any non prime attribute functionally dependent on part of a candidate key.
- PROJ_NAME depends on PROJ_NUM which is part of candidate key. And EMP_Name depends on EMP_NUM. EMP_NUM is also part of candidate key.

Proj_NUM → proj_name

EMP_NUM → EMP_NAME, JOB_CLASS, CHG_HOUR

- Are partial dependency.

- So whole table project_employee is decomposed in way which remove partial dependency.
- To remove partial dependency create new table in which all non prime attribute is fully depend on primary key attribute
- New table will be

PROJECT (PROJ_NUM, PROJ_NAME)

Employee(EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR)

ASSIGN (PROJ_NUM, EMP_NUM, HOURS)

3rd Normal Form (3NF)

- Rules for table /database in 3 NF.

  1 . The tables should be in the second Normal Form 2NF.

  2 . There should be Transitive Dependency.(remove transitive dependency in 3NF)

  Employee(EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR)

  Dependecy form 2NF

  JOB_CLASS is depends on EMP_NUM . EMP_NUM-> JOB_CLASS

  CHR_HOUE is depends on JOB_CLASS.

  So EMP_NUM->JOB_CLASS and JOB_CLASS-> CHG_HOUR

- A -> B B->C A->C- transitive dependency

- So transitive dependency between EMP_NUM and CHR_HOUR
- EMP_NUM-> CHR_HOUR is transitive dependency.
- In 3NF transitive dependency is removed.
- So decompose Employee table to remove transitive dependency.

PROJECT (PROJ_NUM, PROJ_NAME)

ASSIGN (PROJ_NUM, EMP_NUM, HOURS)

as it is

Employee(EMP_NUM, EMP_NAME, JOB_CLASS, CHG_HOUR)

Is decomposed.

- Employee(EMP_NUM, EMP_NAME, JOB_CLASS)
- JOB(JOB_CLASS, CHG_HOUR)
- Employee table decompose and transitive dependency is removed and it create new table JOB

- So final tables in 3NF are

  PROJECT (PROJ_NUM, PROJ_NAME)

  ASSIGN (PROJ_NUM, EMP_NUM, HOURS)
- Employee(EMP_NUM, EMP_NAME, JOB_CLASS)
- JOB(JOB_CLASS, CHG_HOUR)

Boyce-Codd Normal Form (BCNF) 3.5

- A table is in Boyce-Codd normal form (BCNF) if every determinant in the table is a candidate key.
- BCNF is a special case of 3NF.
- If a table contains only one candidate key, the 3NF and the BCNF are equivalent.

An interesting problem

Let us assume the following reality

- For each subject, each student is taught by more than one instructor
- Each instructor teaches only one subject
- Each subject is taught by several instructors

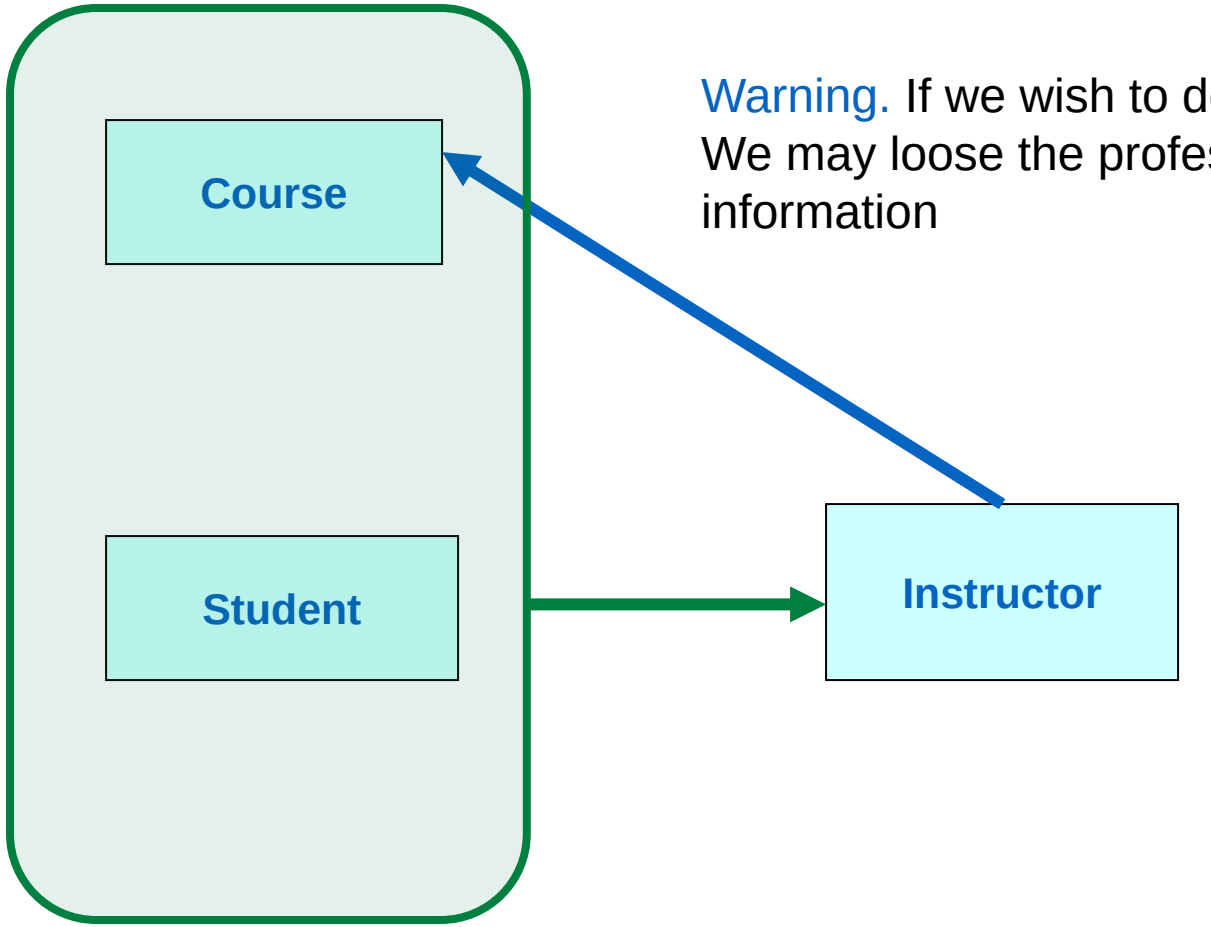
Student	Subject	Teacher
Smith	Math	Dr. White
Smith	English	Dr. Brown
Jones	Math	Dr. White
Jones	English	Dr. Brown
Doe	Math	Dr. Green

An interesting relation

From the course point of view => *School:* (**Course**, **Student,** Instructor)

From the student point of view => *Learning:* (**Student, Instructor,** Course)

From the instructor point of view => *Teaching:* (**Instructor,** Course)

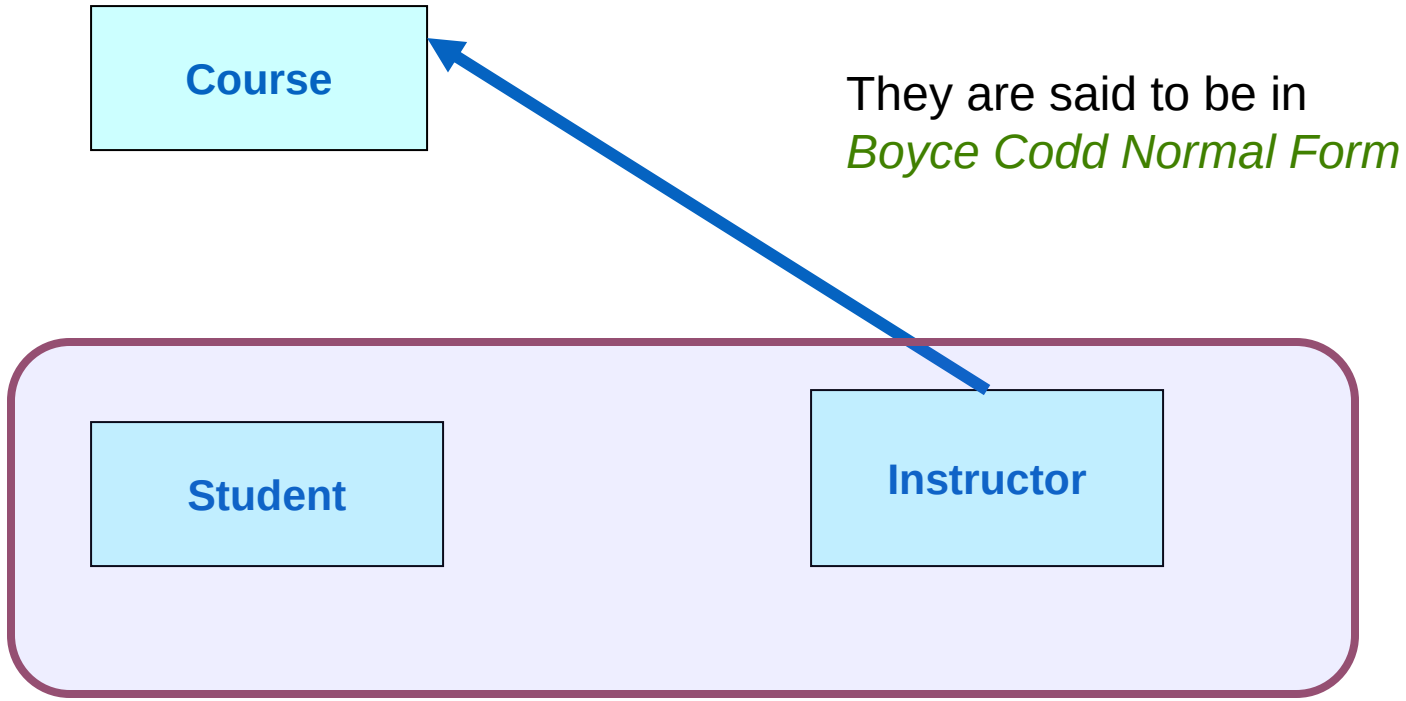


Warning. If we wish to delete a student
We may loose the professor
information

The Solution

From the student point of view => *Learning:* (**Student, Instructor)**)

From the instructor point of view => *Teaching:* (**Instructor,** Course)



- it's clear we have the following functional dependency:
- Teacher -> Subject
- And the left side of this dependency is not the primary key.
- So, to convert the table from 3NF to BCNF, we do these steps:
  - Determine in the table, a key other than the primary key. That can be left side to the functional dependency.
  - Delete the key in the right side of our functional dependency in the main table.
  - Make a table for this dependency, with it's key being the left side of the dependency, as the following:

  - Stud,sub-> teacher A-> B & B->C , bc non prime attribute (transitive)
  - Teacher-> sub
  - stud,teacher teacher subject

Student	Teacher
Smith	Dr. White
Smith	Dr. Brown
Jones	Dr. White
Jones	Dr. Brown
Doe	Dr. Green

Teacher	Subject
Dr. White	Math
Dr. Brown	English
Dr. Green	Math

Multivalued dependency (type of FD)

- Multivalued dependency occurs when two attributes in a table are independent of each other but, both depend on a third attribute.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why it always requires at least three attributes.

CAR_MODEL	MANUF_YEAR	COLOR
M2011	2008	White
M2001	2008	Black
M3001	2013	White
M3001	2013	Black
M4006	2017	White
M4006	2017	Black
M2011	2009	

- Here color is depends on car_model
- Manufacturing year depends on car_model
- Car_model - >> manufacturing_year
- Car_model - >> color

- Color and manufacturing_year is independent form each other.
- So here car_model and manufacturing_year is repeating for different color in first two row.

Fourth Normal Form (4NF)

- Rules for table/database in 4NF

1. Form must be in BCNF
2. There is no multivalued dependencies (remove the multivalued dependencies).

- So to remove the multivalued dependencies of
- Car_detail(car_model , manuf_year, color)
- Manuf_year and color are multivalued attribute and they are independent of each other .
- Car_model and manuf_year detail is repeated to show the different value of color.

- decomposed table in 4NF
- Car_year(car_model,manf_year)
- Car_color(car_model,color);

Example

- The ClientRental relation is defined as follows,
- ClientRental (clientNo, propertyNo, 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

- 1 NF

client(clientno,propertyno,cName,pAddress,rentFinish,rent,ownerNO,oNam
e)

Candidate key : clientno,propertyno

2NF : remove partial dependency

1) cname is depends on only clienno which is part of candidate key.

2) pAddress,rent,ownerNO,oName is are depends on property no.

clientno->clientname)

propertyno -> pAddress,owenerNo,oName,rent)

clientno,propertyno-> rentStart,rentFinsih

2NF

- Tables for 2NF is
- client(clientno,cName)
- Property(propertyno , pAddress,owenerNo,oName,rent)
- Client_rent(clientno,propertyno,rentstart,rentfinish)

- 3 NF

  remove transitive dependency.

  ownerNO-> oName

  properyNo-> ownerNO

So transitive dependency propertyNo -> oName

Remove transitive dependency:

- propertyno->pAddress,onwerNo,rent
- ownerNo->oName

- Tables for 3 NF
  - Property(propertyno , pAddress,owenerNo,rent)
  - Owner(ownerNo,oName)
  - client(clientno,cName)
  - Client_rent(clientno,propertyno,rentstart,rentfinish)

Closer of Function dependency

- The Closure Of Functional Dependency means the complete set of all possible attributes that can be functionally derived from given functional dependency using the inference rules.
- If “F” is a functional dependency then closure of functional dependency can be denoted using “$\{F\}^+$

- Step-1 : Add the attributes which are present on Left Hand Side in the original functional dependency.
- Step-2 : Now, add the attributes present on the Right Hand Side of the functional dependency.
- Step-3 : With the help of attributes present on Right Hand Side, check the other attributes that can be derived from the other given functional dependencies

- E.g
- **Consider the table student_details having (Roll_No, Name,Marks, Location) as the attributes and having two functional dependencies.**
- **FD1 : Roll_No ->Name, Marks**
- **FD2 : Name -> Marks, Location**
  - Closer of Roll_no
  - {roll_no}+ = roll_no,Name,marks
  - {roll_no}+ = roll_no,Name,marks,Location -> closer of roll_no
  - Name+ = name,marks,location
  - IF closer of attribute contains all the attributes then that attribute can be a one of the candidate key.

- Closer of Mark
  - Marks is not on left hand side so no closer for it
- Closer of Name
  - {Name}+ = Name,Marks, Location
- E.g : **Consider a relation R(A,B,C,D,E) having below mentioned functional dependencies. Find out the closer of each attribute.**
- **FD1 : A ->BC**
- **FD2 : C -> B**
- **FD3 : D -> E**
- **FD4 : E -> D**

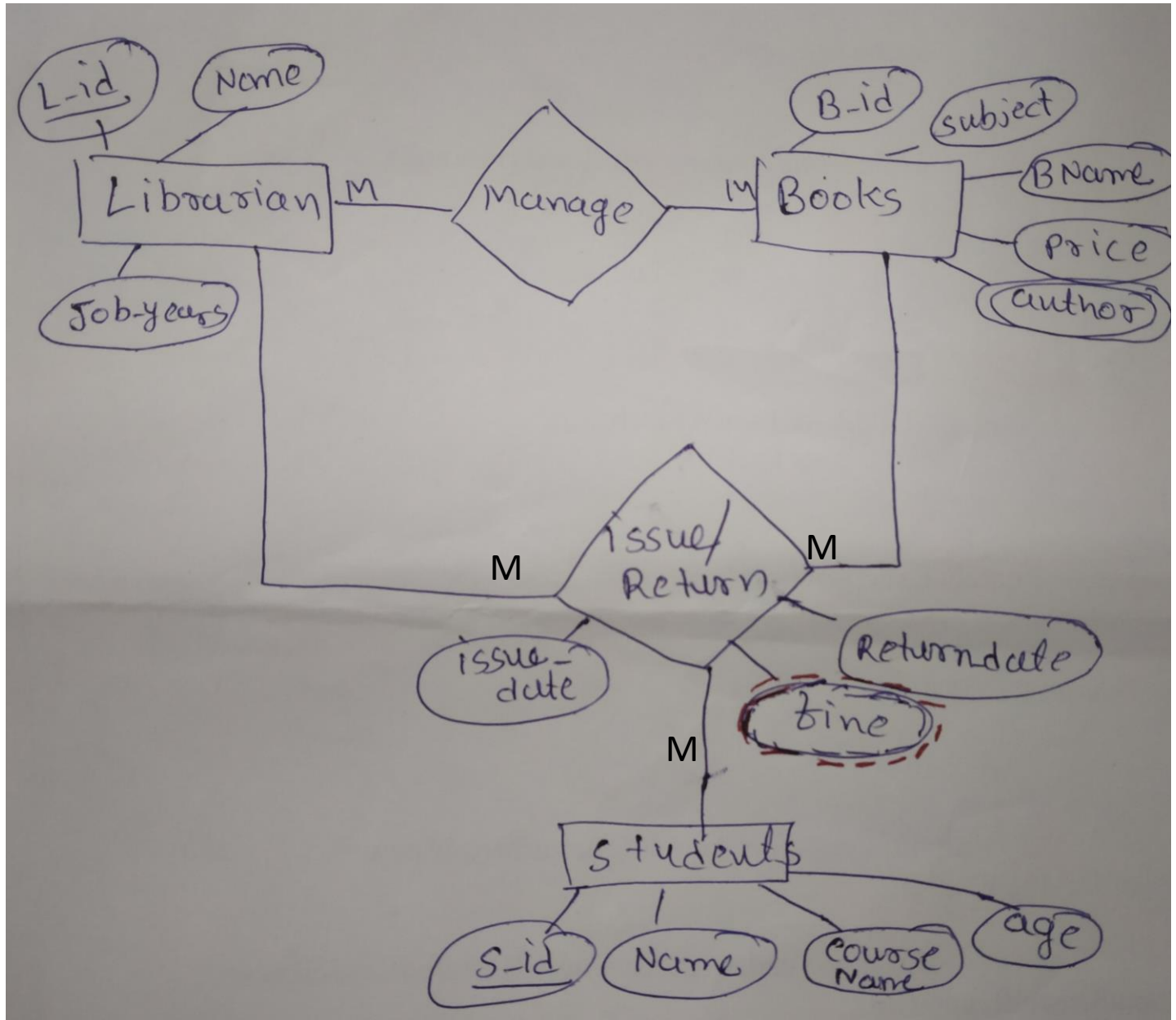
- {A}+ = {A,B,C}
- {B}+= NIL
- {C}+ = {B,C}
- {D}+ = {D,E}
- {E}+ = {E,D}

- As any of closer of the key not derived all attribute so it can be derived form the combine two or more attribute

- {AD}+ = {A,B,C,D,E}
- {AE}+ = {A,B,C,D,E} so AD and AE can be candidate key for given functional dependency.

- Library management system
  - Book can be issue/return by students.
  - students can issue many books.
  - Librarian can issue/returns books to the Students.
  - Fine is also maintain based on issue and return books.
  - Librarian can manage books (add ,delete book data)

  - Book,student,librarian : entitiy
  - Issue/return , manage : relation


L_id
Name
Librarian
M
Manage
M
B_id
subject
Books
BName
Price
Author
Job-years
M
Issue/
Return
M
issue_
date
Returndate
fine
M
Students
S_id
Name
Course
Name
age