Cauvery College for Women (Autonomous)

Nationally Accredited (III Cycle) with ‘A’ Grade by NAAC

Annamalai Nagar, Tiruchirappalli-18.

Name of Faculty	:Ms.Lakshna Arun
Designation	: Asst Professor
Department	: Computer Applications
Contace Number	: 9487936636
Programme	: BCA
Batch	: 2018-2021
Semester	: IV
Course	: Database Management Systems
Course Code	:16SCCCA4

Unit : V

Topics Covered : Relational database design,1NF,2NF

Relational Database Design

- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Functional Dependency Theory
- Algorithms for Functional Dependencies
- Decomposition Using Multivalued Dependencies
- More Normal Form
- Database-Design Process

The Banking Schema

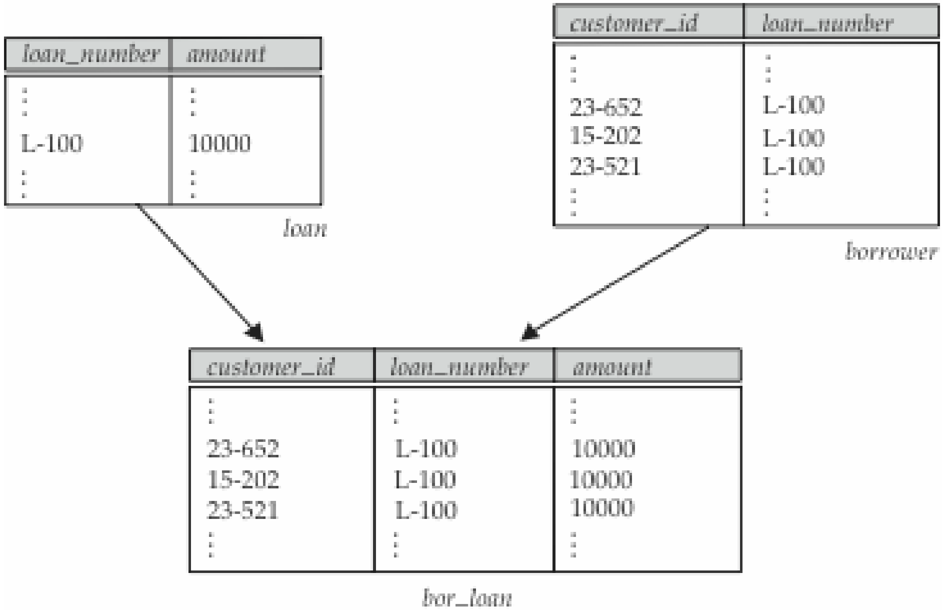
- branch = (branch_name, branch_city, assets)
- customer = (customer_id, customer_name, customer_street, customer_city)
- loan = (loan_number, amount)
- account = (account_number, balance)
- employee = (employee_id. employee_name, telephone_number, start_date)
- dependent_name = (employee_id, dname)
- account_branch = (account_number, branch_name)
- loan_branch = (loan_number, branch_name)
- borrower = (customer_id, loan_number)
- depositor = (customer_id, account_number)
- cust_banker = (customer_id, employee_id, type)
- works_for = (worker_employee_id, manager_employee_id)
- payment = (loan_number, payment_number, payment_date, payment_amount)

Combine Schemas

✓Suppose we combine *borrower* and *loan* to get

bor_loan = (*customer_id*, *loan_number*, *amount*)

✓Result is possible repetition of information (L-100 in example below)



loan_number	*amount*
⋮	⋮
L-100	10000
⋮	⋮

loan

customer_id	*loan_number*
⋮	⋮
23-652	L-100
15-202	L-100
23-521	L-100
⋮	⋮

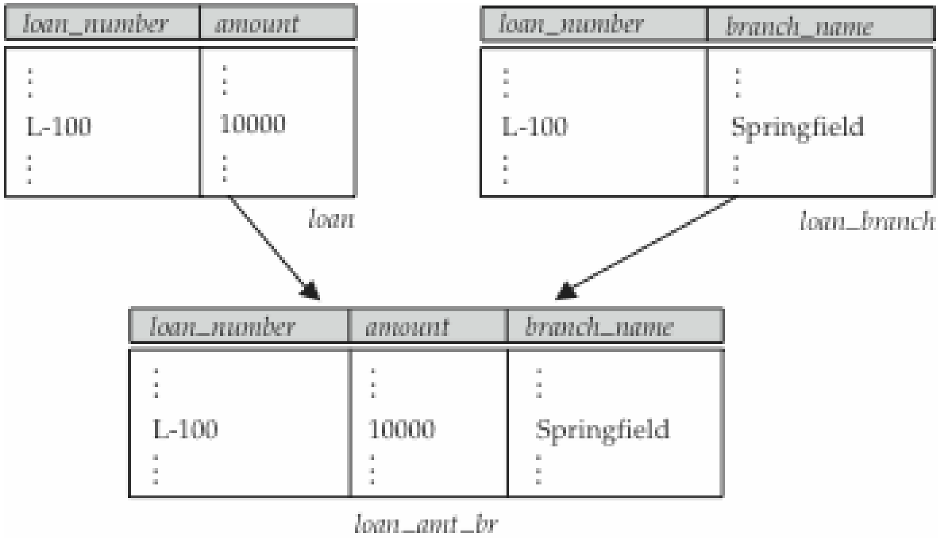
borrower

customer_id	*loan_number*	*amount*
⋮	⋮	⋮
23-652	L-100	10000
15-202	L-100	10000
23-521	L-100	10000
⋮	⋮	⋮

bor_loan

Combined Schema Without Repetition

- ✓Consider combining *loan_branch* and *loan loan_amt_br* = (*loan_number*, *amount*, *branch_name*)
- ✓No repetition (as suggested by example below)



loan_number	*amount*
⋮	⋮
L-100	10000
⋮	⋮

loan

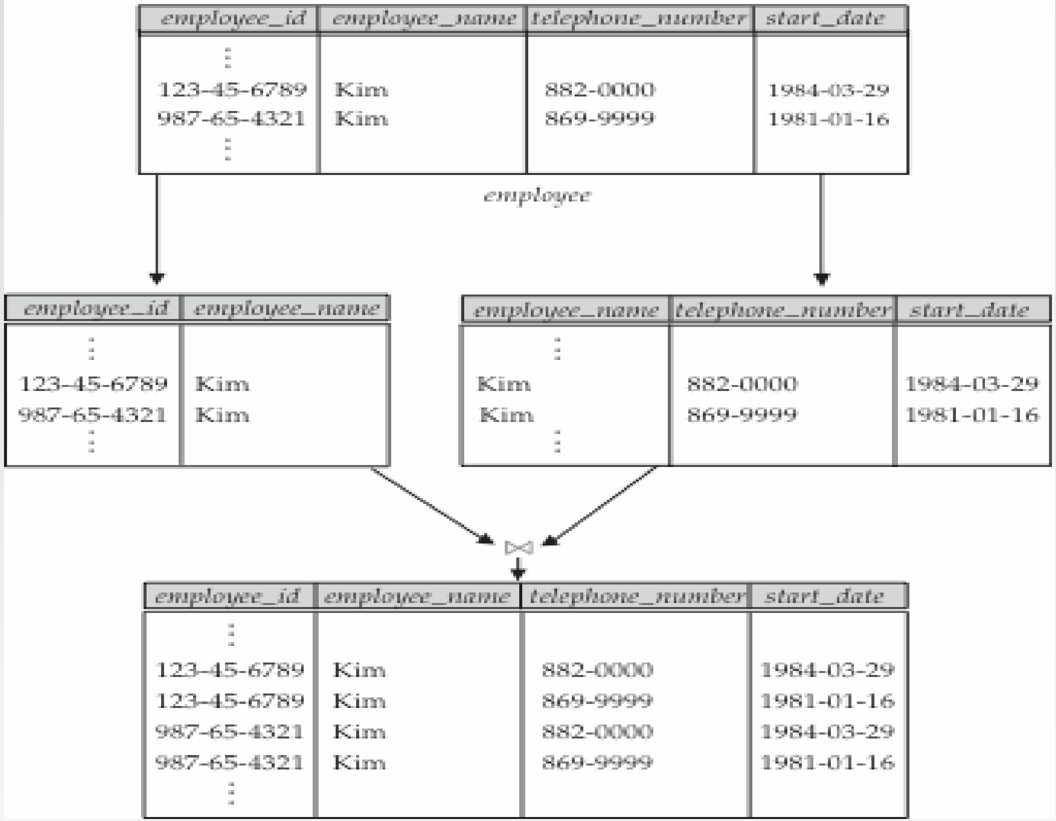
loan_number	*branch_name*
⋮	⋮
L-100	Springfield
⋮	⋮

loan_branch

loan_number	*amount*	*branch_name*
⋮	⋮	⋮
L-100	10000	Springfield
⋮	⋮	⋮

loan_amt_br

A Lossy Decomposition



employee_id	*employee_name*	*telephone_number*	*start_date*
⋮			
123-45-6789	Kim	882-0000	1984-03-29
987-65-4321	Kim	869-9999	1981-01-16
⋮			

employee

employee_id	*employee_name*
⋮	
123-45-6789	Kim
987-65-4321	Kim
⋮	

employee_name	*telephone_number*	*start_date*
⋮		
Kim	882-0000	1984-03-29
Kim	869-9999	1981-01-16
⋮		

$\bowtie$

employee_id	*employee_name*	*telephone_number*	*start_date*
⋮			
123-45-6789	Kim	882-0000	1984-03-29
123-45-6789	Kim	869-9999	1981-01-16
987-65-4321	Kim	882-0000	1984-03-29
987-65-4321	Kim	869-9999	1981-01-16
⋮			

First Normal Form

✓Domain is atomic if its elements are considered to be indivisible units Examples of non-atomic domains:

- □Set of names, composite attributes
- □Identification numbers like CS101 that can be broken up into parts

✓A relational schema R is in first normal form if the domains of all attributes of R are atomic

✓Non-atomic values complicate storage and encourage redundant (repeated) storage of data

- □Example: Set of accounts stored with each customer, and set of owners stored with each account

Functional Dependencies

✓*A* functional dependency is trivial if it is satisfied by all instances of a relation

□Example*:*

□*customer_name*, *loan_number* ® *customer_name*

□*customer_name* ® *customer_name*

□In general, a ® *b* is trivial if *b* Í a

Example

✓*R* = (*A*, *B*, *C*)

F = {*A* ® *B*, *B* ® *C*)

□Can be decomposed in two different ways

✓*R*1 = (*A*, *B*),*R*2 = (*B*, *C*)

□Lossless-join decomposition:

*R*1 Ç *R*2 = {*B*} and *B* ® *BC*

□Dependency preserving

✓*R1* = (*A*, *B*), *R2* = (*A*, *C*)

□Lossless-join decomposition:

*R*1 Ç *R*2 = {*A*} and *A* ® A*B*

□Not dependency preserving

(cannot check *B* → *C* without computing R_1 R_2)