module 1 :

→ 2 classes are ~~there~~ present

i) Attr

ii) AssignModule 1.

↳ This contains the main function & Scanner class.

→ class "Attr" contains all the stuff.

→ This class contains

i) 15 variables. (excluding method specific variables).

ii) 22 methods.

Algorithm;

I have divided the work into 5 parts.

i) computing FD closure

ii) computing minimal keys.

iii) computing Normal Form of each FD.

iv) Decomposing.

v) Testing for Dependency preserving.

I) computing FD closure;

- we are using "iR_6" to compute FD closure because it includes iR_3 with in itself. and "iR_6" is having more information when compared with iR_3.

iR_1 –	$A \to A$
iR_2 –	$x \to y \Rightarrow xz \to yz$
iR_3 –	$x \to y$ & $y \to z \Rightarrow x \to z$
iR_4 –	$x \to yz \Rightarrow x \to y$ & $x \to z$
iR_5 –	$x \to A$, $x \to B \Rightarrow x \to AB$
iR_6 –	$x \to y$ & $wy \to z$ then $wx \to z$

→ iR_1, iR_2, iR_4, iR_5 are anyways useless.

→ In order to understand how it works, PTO.

write all the FDs in 2 columns like this...

(i)	(j)
AB → C	AB → C
C → D	C → D
CE → FD	CE → FD
B → H	B → H

use 2 for loops (nested) — one for "i" & another for 'j'

& check whether ~~we~~ iR_6 is applicable.

ie., rhs[i] should ~~contain~~ be present in lhs[j]

then we can apply iR_6.

Eg:

AB → C & CE → FD.

⇒ ABE → FD. → new FD created.

append it ~~at the bottom~~.

check other way round also.

ie.)

if rhs[j] ~~should~~ ~~to~~ is present in lhs[i]

then also we can apply iR_6.

used methods;

o) fillFDs.

i) public void fd closure

ii) sort

iii) remove Duplicates

iv) removeRedundant FDs (overloaded once)

v) remove null spaces.

---

2) computing minimal keys;

i) To compute this, FD closure should be ready.

ii) All $2^n - 2$ combinations of attributes should also be ready.

→ ~~we store~~ we first store all these $(2^n - 2)$ combinations in a String[] called as "attr-combns".

→ To compute all the $(2^n - 2)$ combinations, I used Recursion concept.

→ methods used are

i) fillAttrCombns.

ii) nChooseR

iii) store

→ computing keys :

→ ~~Take every combination~~

→ Take appropriate combination from "attr_combns" & check whether it is a key.

→ Don't check for all combinations.

ie, if AB is already a key then checking for ABC, ABD, ABCDE is waste of time

store the foundkeys in "keys" array..

Used methods;

i) fillkeylist

ii) getCombination()

iii) Computing Normal form of Each FD;

Basically Every FD belong to any of the following 9 cases. (Assuming only one attribute on RHS).

i) key → key (BCNF)

eg: AB → C (let us say both are keys)

ii) key → Another key's Attribute (BCNF)

eg: ∃ 2 keys
AB, DE

& AB → D.

iii) key → Non key attribute (BCNF)

E

iv) Keyattribute → key

eg: ∃ 2 keys AB, C

B → C

Impossible case.

Because if keyattribute → key then that keyattr becomes key

v) keyattribute → keyattribute (3NF)

Eg: ∃ 2 keys i) ABC
ii) DE

E → C

vi) keyattribute → non keyattribute (1NF)

Eg: ∃ a key AB.

A → C

vii) non-keyattribute → key (impossible case).

viii) non-keyattribute → key attribute (3NF).

Eg: ∃ a key ABC

D → B

ix). non-keyattribute → non key attribute (2NF).

In order to calculate NF, take a FD, scan it for the above 9 cases & find out the Normal form. & put it in nf_list[ ]

Note:

out of 9,

- 2 impossible
- 1 2NF case → can be decomposed to BCNF
- → 1 1NF case → " " "
- → 2 3NF cases →

v) can't be decomposed with dependency preserving
ie, this is final

viii) can be brought to BCNF by decomposition.

→ 3 BCNF.

used Methods:

i) fillNFList()

ii) isAKey()

iii) isAKeyAttribute()

v) iv) Decomposing:

→ from the above NF,

→ decompose..

used methods:

→ decomposeRel()

→ completeFormalities()

↳ To compute keys & FDs for newly formed relation

---

v) Testing for Dependency preserving:

i) Take FD closures from all the decomposed rels and store in all of them a temporary String[].

ii) calculate FD closure.

iii) count the number of FDs from initial & final cases.

iv) compare every FD.

used method;

i) calculateFDClosureFromDecomposedAndCompareWithOriginal().

Total methods ~~used~~ explained till now: ~~1+1~~ 17+1.

Remaining methods:

i) Public void printrel()

ii) Public void printNF()

iii) Printkeys

iv) Print FDs.