Closure of Attribute Sets

Given a set α of attributes of R and a set of functional dependencies F, we need a way to find all of the attributes of R that are functionally determined by α. This set of attributes is called the **closure of α under F** and is denoted α+. Finding α+ is useful because:

- if α+ = R, then α is a superkey for R
- if we find α+ for all α⊆ R, we've computed F+ (except that we'd need to use decomposition to get all of it).

An algorithm for computing α+:

```
result := α
repeat
   temp := result
   for each functional dependency  β → γ  in F do
      if  β ⊆  result then
         result := result ∪ γ
until temp = result
```

Problem:

Compute the closure for relational schema
R={A,B,C,D,E}
A-->BC
CD-->E
B-->D
E-->A
List candidate keys of R.

Solution:
R={A,B,C,D,E}
F, the set of functional dependencies **A-->BC, CD-->E, B-->D, E-->A**

Compute the closure for each β `in` β → γ `in F`

Closure for A

Iteration	result	using
1	A	
2	ABC	A-->BC
3	ABCD	B-->D
4	ABCDE	CD-->E
5	ABCDE	

A+ = ABCDE, Hence A is a super key

Closure for CD

Iteration	result	using
1	CD	
2	CDE	CD-->E
3	ACDE	E-->A
4	ABCDE	A-->BC
5	ABCDE	

CD+ = ABCDE, Hence CD is a super key

Closure for B

Iteration	result	Using
1	B	
2	BD	B-->D
3	BD	

B+ = BD, Hence B is NOT a super key

Try applying Armstrong axioms, to find alternate keys.

B-->D
BC-->CD (by Armstrong's augmentation rule)

Closure for BC

Iteration	result	using
1	BC	
2	BCD	BC-->CD
3	BCDE	CD-->E
4	ABCDE	E-->A

BC+ = ABCDE, , Hence BC is a super key

Closure for E

Iteration	**result**	**using**
1	E	
2	AE	E-->A
3	ABCE	A-->BC
4	ABCDE	B-->D
5	ABCDE	

E+ = ABCDE

A and E are minimal super keys.
To see whether CD is a minimal super key, check whether its subsets are super keys.
C+ = C
D+ = D
Since C and D are not super keys, CD is a minimal super key.

To see whether BC is a minimal super key, check whether its subsets are super keys.
B+ = BD
C+ = C
Since B and C are not super keys, BC is a minimal super key.

Since A, BC, CD, E are minimal super keys, they are the candidate keys.

A, BC, CD, E

If there are 5 attributes, then we need to check 32 (2^5) combinations to find all super keys. Since we are interested only in the candidate keys, the best bet is to check closure of attributes in the left hand side of functional dependencies.

If the closure yields the relation R, it is super key. Check whether it is a minimal super key, by checking closure for its subsets.

If the closure didn't yield the relation R, it is not a super key. Try applying Armstrong's axioms, to get an attribute combination that is a super key. Check to see it also an minimal super key.

The list of minimal super keys obtained is the candidate keys for that relation.

A **superkey** is a set of one or more attributes that allow entities (or relationships) to be uniquely identified.

Examples for the given problem:

A,CD, E, BC, AE, AB, ABE,ACD,BCD, DE etc.

(Any attribute added with the minimal super keys A, CD, E is also a super key).

A **candidate key** is a superkey that has no superkeys as proper subsets. A candidate key is a minimal superkey.

Examples for the given problem:

A, BC, CD, E

The **primary key** is the (one) candidate key chosen (by the database designer or database administrator) as the primary means of uniquely identifying entities (or relationships).

Example for the given problem:

Any one of the above three (A, BC, CD, E) chosen by the database designer.