PART 1:
FUNCTIONAL DEPENDENCIES

Functional dependencies

- Let **X**, **Y** be sets of attributes from relation **R**
- **X -> Y** (we say: "**X** *functionally determines* **Y**")
  - Any tuples in **R** which agree in all attributes of **X** must also agree in all attributes of **Y**
- X → A ("X determines A") is an assertion about a relation R:
  - whenever two tuples of R agree on all the attributes of X, then they must also agree on the attribute A
  - t1[X] = t2[X] implies t1[A] = t2[A] for all t1, t2 in R (analogously for X → Y)

Functional dependencies

Roll_No	Student_Name	Dept_Name	Dept_Building
2	abc	CS	A4
3	pqr	IT	A3
4	xyz	CS	A4
5	xyz	IT	A3
6	mno	EC	B2
7	jkl	ME	B2

Functional dependencies

Table Drinkers(name, addr, beersLiked, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

Functional dependencies

Table Drinkers(name, addr, beersLiked, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Janeway	Voyager	Bud	A.B.	WickedAle
Janeway	Voyager	WickedAle	Pete's	WickedAle
Spock	Enterprise	Bud	A.B.	Bud

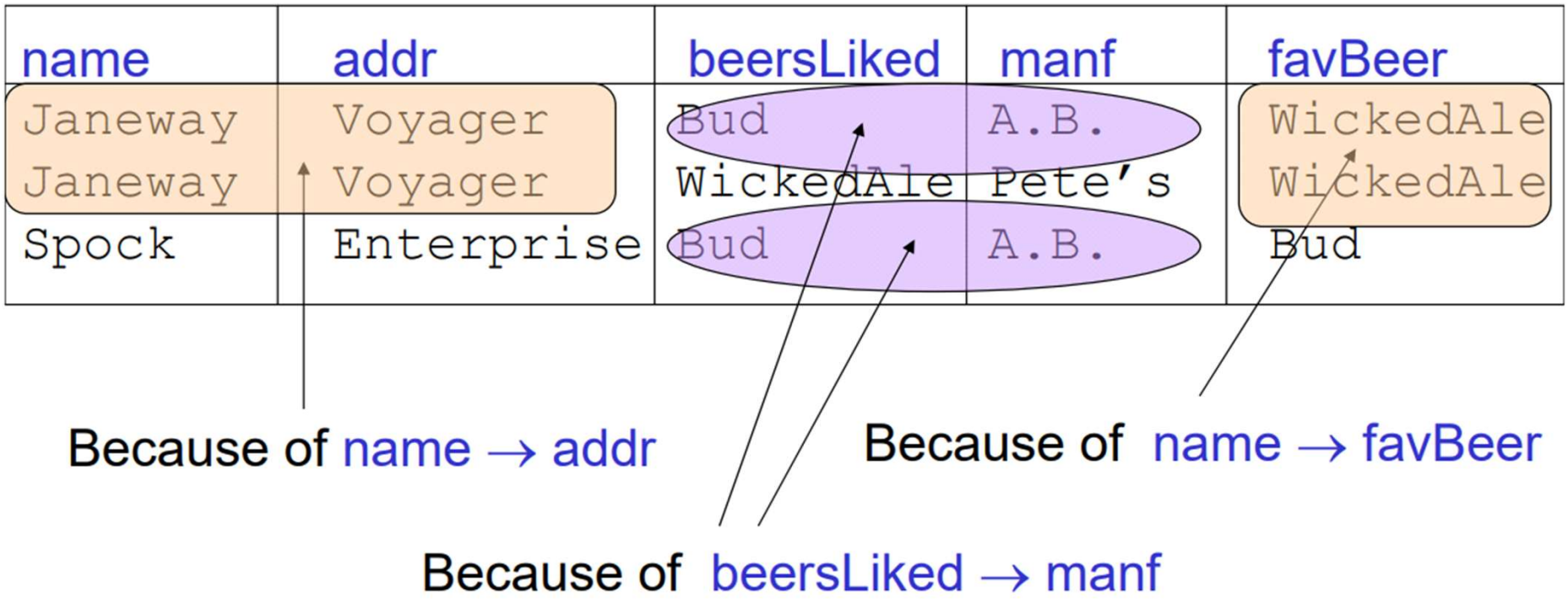
Reasonable FD's to assert:

1.name → addr

2.name → favBeer

3.beersLiked → manf

Functional dependencies



FD

Given the following attributes, identify all the functional dependencies:

- shipment_id
- shipment_date
- origin
- destination
- ship_id
- ship_name
- captain_id
- captain_name
- item_id
- description
- weight
- quantity

Rules and principles about FDs

- Rules
  - The splitting/combining rule
  - Trivial FDs
  - The transitive rule
- Algorithms related to FDs
  - the closure of a set of attributes of a relation

The Splitting/Combining rule of FDs

- Attributes on right independent of each other
  - Consider *a,b,c -> d,e,f*
  - "Attributes a, b, and c functionally determine d, e, and f"

  => No mention of d relating to e or f directly
- Splitting rule (useful to split up right side of FD)
  - *abc -> def* becomes *abc -> d*, *abc -> e* and *abc -> f*
- No safe way to split left side
  - *abc -> def* is NOT the same as *ab -> def* and *c -> def*!
- Combining rule (useful to combine right sides):
  - *if abc -> d*, *abc -> e*, *abc -> f* holds, then *abc -> def* holds

Splitting FDs – example

- Consider the relation and FD
  - EmailAddress(user, domain, firstName, lastName)
  - user, domain -> firstName, lastName
- The following hold
  - user, domain -> firstName
  - user, domain -> lastName
- The following do NOT hold!
  - user -> firstName, lastName
  - domain -> firstName, lastName

Trivial FDs

- Not all functional dependencies are useful
  - *A -> A* always holds
  - *abc -> a* also always holds (right side is subset of left side)
- FD with an attribute on both sides is "trivial"
  - Simplify by removing **L** ∩**R** from **R**

    abc -> ad becomes *abc -> d*
  - Or, in singleton form, delete trivial FDs

    abc -> a and *abc -> d* becomes just *abc -> d*

Transitive rule

- The transitive rule holds for FDs
  - Consider the FDs: *a -> b* and *b -> c*; then *a->c* holds
  - Consider the FDs: *ad -> b* and *b -> cd*; then *ad->cd* holds or just *ad->c* (because of the trivial dependency rule)