FD: 1. Collect all attributes into one R.
2. Specify the FD's and MVD's

All FD's are MVD's!

FD: $A \rightarrow B$ for each A only one B.
MVB: $A \twoheadrightarrow B$ A implies B but B is not unique.

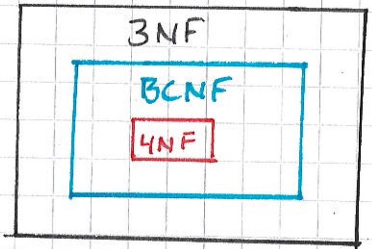
3. Key find. Key≡ $X \rightarrow S$ all attr of S determined by X.
4. Calculate vialations.
5. Decompose (may have new vialations)

BCNF: A FD $X \rightarrow A$ violates BCNF if
- X is not a superkey
- The dependency is not trivial.

3NF : A FD $X \rightarrow A$ violates 3NF if
- X is not a superkey
- The dependency is not trivial
- A is not prime. (A is prime if belong to key

4NF : A MVD $X \twoheadrightarrow A$ violates 4NF if
- X is not a superkey
- The mvd is not trivial.

Property	3NF	BCNF	4NF
Eliminates redundancy due to FD's.	NO	YES	YES
Eliminates redundancy due to MVD's.	NO	NO	YES
Preserves FD'S	YES		
Preserves MUD'S	NO	NO	NO



Examples

BCNF:

country	currency	value
SWE	SEK	0.12
FIN	EUR	1.10
EST	EUR	1.10

→

country	currency
SWE	SEK
FIN	EUR
EST	EUR

+

currency	value
SEK	0.12
EUR	1.10

(Lossless join)

3NF :

City	street	code
GBG	F-gatan	41
GBG	R-vägen	42
GBG	H-vägen	42
Sthlm	B-gatan	13

City, street → Code
code → City
City is part of key => it's okay in 3NF.

4NF: (Symmetrical)

country $\twoheadrightarrow$ product
country $\twoheadrightarrow$ exTO

country	Product	exTO
SWE	cars	NWG
SWE	cars	DEN
SWE	paper	NWG
SWE	Paper	DEN

→

country	product
SWE	cars
SWE	paper

+

country	exTO
SWE	NWG
SWE	DEM

Closure : 1. $X^+ = X$
2. $N = \{A \mid A \in S, A \notin X^+, A \text{ follows from } X^+\}$
3. If $N = \emptyset$, return X^+. Else set $X^+ = X^+ \cup$ New

4NF: 1. If R has no 4NF violations return R.
2. If R has a violating MVD decompose R to
R_1 w sign $X \cup \{Y\}$
R_2 w sign $S - Y$
3. Apply 1 and 2 to R_1 and R_2.

BCNF: 1 If R has No BCNF violations, return R
2. If R has a violating FD decompose R to
R_1 w signature $X \cup \{A\}$
R_2 w signature $S - \{A\}$
3. Apply 1 and 2 to R_1 and R_2.

3NF: 1. If R has no 3NF violations, return R.
2. If R has 3NF violations
- Compute a minimel besis F- of F
- Group F- by the left hand side.
- For each groups return the schema w the common LHS and all the RHSs.
- If one of the schemas contains a key of R these groups are enough, otherwise add a shcema containing just some key.

3NF ex: country currency value
country → currency
country → value
idates 3NF → currency → value
↓ minimal basis
country → currency
currency → value
↓ 3NF
country currency
currency value

Hso BCNF in
his case.

3NF but NOT BCNF

Court	Start	end	rate type
1	09:30	10:30	SAVER
1	11:00	12:00	SAVER
1	14:00	15:30	STANDARD
2	10:00	11:30	B
2	11:30	13:30	B
2	15:00	16:30	A

SAVER: court 1, member
STANDARD: court 1, not member
A : court 2, member
B : court 2, !member

!BCNF: Ratetype → court

BCNF but !4NF

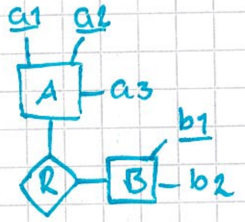
Restaurant	Pizza	Area
A1	Thick	Spring
A1	Thick	Shelby
A1	Thick	Cap
A1	Stuff ~~Thick~~	Spring
A1	Stuffed	Shelby
A1	Stuffed	Cap
Elite	Stuffed	Cap
Elite	Thin	Cap
V	Thick	Spring
V	Thick	Shelby
V	Thin	Spring
V	Thin	Shelby

Only key: {Restaurant, Pizza, Area}

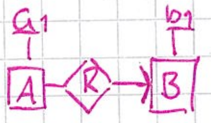
MVD's: Restaurant ↠ Pizza
Restaurant ↠ Area

Relational Algebra:

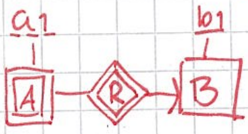
σ_{cond}	= WHERE
π_{proj}	= SELECT
ρ_{name}	= AS
γ_{attr}	= GROUP BY
τ_{exp}	= SORT BY
δ	= DISTINCT
$\times$	= FROM / CROSS JOIN
$\bowtie$	= NATURAL JOIN
$\bowtie_c$	= JOIN ON (theta)
$\bowtie_{att}$	= INNER JOIN
$\bowtie^{o}_{att}$	= FULL OUTER JOIN
$\bowtie^{OL}$	= LEFT — " —
$\bowtie^{OR}$	= RIGHT — " —



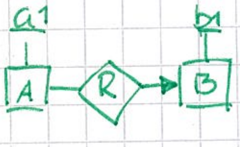
A(<u>a1</u>, <u>a2</u>, a3)
B(<u>b1</u>, b2)
R(<u>a1</u>, <u>a2</u>, <u>b</u>)
(a1, a2) → A.(a1, a2)
b → B.b1



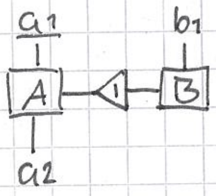
A(<u>a1</u>, b)
b → B.b1
B(<u>b1</u>)



A(<u>a1</u>, <u>b</u>)
b → B.b1
B(<u>b1</u>)



A(<u>a1</u>)
B(<u>b1</u>)
R(<u>a</u>, b)
a → A.a1
b → B.b1



A(<u>a1</u>, a2)
B(<u>a1</u>, b1)
a1 → A.a1

	dirty reads	non-rep read	Phantoms
Serial	–	–	–
repeatable r	–	–	+
read commit	–	+	+
read uncomm	+	+	+ ← FINNS EJ PSQL

Dirty read: Read data from concurrent-uncommited-transaction.

Non-rep : Read data twice and get diff results, because update! or delete

Phantoms : Same Query => diff results, because insertion.

```
<word>
   <pos> Noun </Pos>
      <english> computer </english>
      <Swedish> dater </swedish>
</word>
```

DTD1

```
<!Element word(pos, english, swedish)>
<!Element pos (#PCDATA)>
<!Element english(#PCDATA)>
<!Element swedish(#PCDATA)>
```

DTD2

```
<!Element word EMPTY>
<!Attlist word
    POS CDATA #REQUIRED
    english CDATA #REQUIRED
    swedish CDATA #REQUIRED
>
      ⇓
<word pos="noun" english=
 "computer" swedish="dater">
```

```
CREATE TRIGGER trigger when FOR EACH ROW|STATEMENT EXECUTE PROCEDURE f();
when := BEFORE|AFTER event ON tablename
        |INSTEAD OF event ON viewname
event:= INSERT |UPDATE|DELETE
CREATE FUNCTION f() RETURNS TRIGGER AS $$
   BEGIN
     *
   END
   $$ LANGUAGE 'plpgsql';
INSERT INTO table(attrib+) values(values+);
CREATE VIEW name AS (query);
CREATE TABLE name(  );
```

Indexes:	W	W/O
Read all tup:	1	n
Lookup :	1	n
Modification:	4	2n
Insert :	4	2
Fetch all :	1+k, k<<n	
Fetch ind=key:	2(1+k), k=1	