What is the minimum total quantity of all orders for a product?

```
select min(sum(Quantity))
from orders
group by Prodname;
```

```
select Prodname, sum(Quantity)
from orders
group by Prodname;
```

⇒ P1 6 ⇒ min(sum Quantity) ⇒ 4
P2 4 ⇒
P3 7 ⇒

orders

PN	Qty	..
P1	1	
P1	2	
P1	3	
P2	2	
P2	2	
P3		
P3	1	
P3		

6 ; 4 ; 7

Nov 5th:

```
create view PC_SUPPLS as
select SName, SAddress, Chain
from SUPPLIERS S
where exists (select * from offers
              where SName = S.SName
                and Prodname = 'MegaPC');
```

"virtual view"

PC_Suppls

suppliers ⋈ offers

insert
update
delete

updates to base relations are automatically reflected in the view

materialized view

insert into Mat_PC_Suppls
(...Query...)

are not automatically reflected

Modifications of a View

- Consider the view

  CUST_ORDERS(FName, LName, Prodname, SName, Quantity)

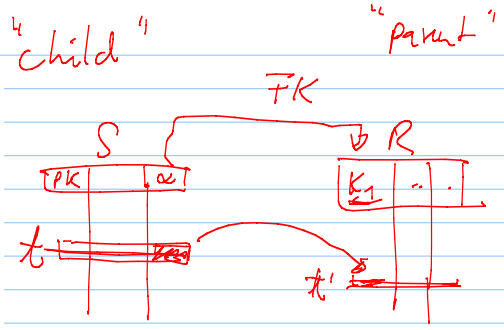
  defined as

  select C.FName, C.LName, Prodname, SName, Quantity
  from CUSTOMERS C, orders O
  where C.FName=O.FName **and** C.LName=O.LName;

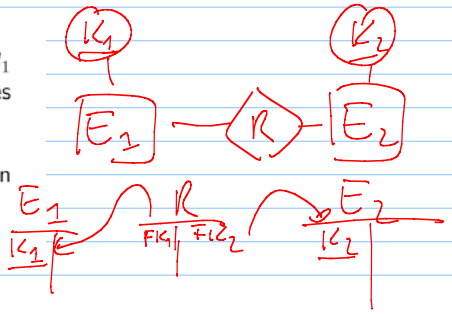
view table
↓
CUST_ORDERS ⟵
CUSTOMERS ⋈ ORDERS
base tables

view def.

- Formal definition:
  - Let $r(R)$ and $s(S)$ be relations with primary keys K_1 and K_2 respectively.
  - The subset α of attributes of S is a *foreign key* referencing K_1 in r, if for every tuple t in s there must be a tuple t' in r such that $t'[K_1] = t[\alpha]$.
  - Referential integrity constraint: $\pi_\alpha(S) \subseteq \pi_{K_1}(R)$



- Referential Integrity in the ER Model:
  - Consider a relationship set R between two entity sets E_1 and E_2. The relation schema corresponding to R includes the primary keys K_1 of E_1 and K_2 of E_2.
  - Then K_1 and K_2 form the foreign keys to the relation schemas for E_1 and E_2, respectively.



Assume referential integrity constraint $\pi_\alpha(S) \subseteq \pi_K(R)$

IC_1 $\pi_{\text{Prodname}}(\texttt{offers}) \subseteq \pi_{\text{Prodname}}(\texttt{PRODUCTS})$ or

IC_2 $\pi_{\text{FName, LName}}(\texttt{orders}) \subseteq \pi_{\text{FName, LName}}(\texttt{CUSTOMERS})$

offrs.Prodname FK→ PRODUCTS.Prodname

Critical operations

insert into offers

delete from PRODUCTS

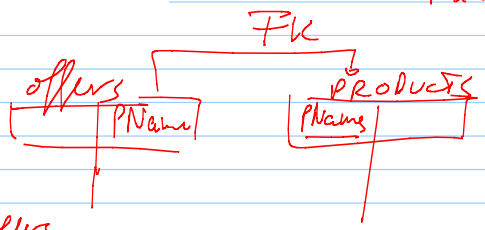
update offers

update PRODUCTS

not critical

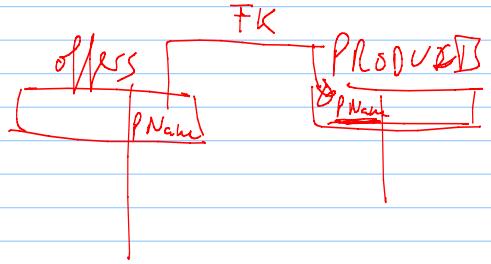
insert into PRODUCTS

delete from offers




```
create table offers (
   Prodname   varchar(80) references PRODUCTS
                          on delete cascade,
. . . );
```

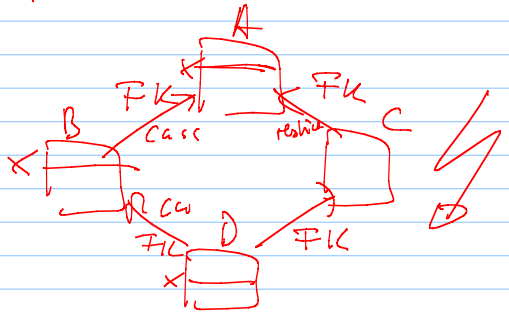
deletion of the parent row, cascades to the children that reference it.



on delete cascade

on delete restrict

on delete No action



avoid this!

For each product, there must be at least two suppliers.

```
create assertion two_suppliers check
(not exists (select * from offers O1
             where not exists
                   (select * from offers O2
                    where O1.SName <> O2.SName
                      and O1.Prodname = O2.Prodname))
)
```

ok(P) ←
offers(P, S_1),
offers(P, S_2),
$S_1 \neq S_2$.

ic_violation_2supp(P) ← product(P, ...),
not ok(P)

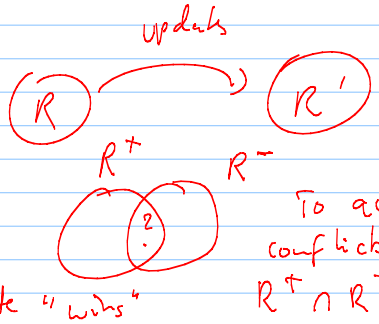
---

create trigger <name>
{**before**|**after**} <trigger event(s)>
on <table> [**referencing** <transition table or variable list>]
[**for each** {**row** | **statement**}]
[**when** <condition>]
<triggered SQL statement>

E
C
A
action

A trigger is *fired* if <trigger event(s)> occurred before/after an event in a transaction (immediate/deferred);
A trigger is *executed* if <condition> evaluates to true.

transition-Δ-tables R^+, R^-
R^{ins}, R^{del}
R^{old}, R^{new}



The modified relation R' after transaction T thus can be obtained as
$R' = (R - R_{\text{deleted}}) \cup R_{\text{inserted}} - R_{\text{updated\_old}} \cup R_{\text{updated\_new}}$.

insert "wins"

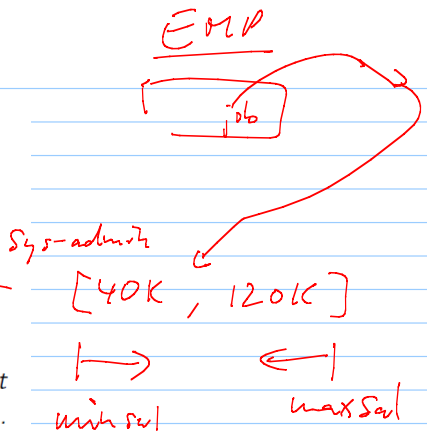
$R' = (R \cup R^+) - R^-$
delete "wins"

To avoid conflicts
$R^+ \cap R^- = \emptyset$

EMP(Empno, EName, Job → SALGRADE, Mgr, Hiredate
Sal, Deptno → DEPT)

DEPT(Deptno, Dname, Loc, Budget)

SALGRADE(Job, Minsal, Maxsal)



Let's see how to implement the following integrity constraint:

"The salary of an employee different from the president cannot be decreased and must also not be increased by more than 10%. Furthermore, depending on the job title, each salary must lie within a certain salary range."

This constraint might be affected by operations on EMP and SALGRADE, so we need *two* triggers. . .

- SQL
- Relational Algebra (RA)
- Relational Calculus (RC)
- **Datalog**

SELECT ... FROM ... WHERE ...

$\sigma, \pi, \bowtie, \delta, \cup, \setminus$

$\forall x F,\ \exists x F,\ F \wedge G,\ F \vee G, \neg F$

$\approx$ RC + Recursion

EXAMPLE: Given relations employee(Emp, Salary, DeptNo) and dept(DeptNo, Mgr), find all (employee, manager) pairs:

- SQL:
```
SELECT Emp, Mgr
FROM employee, dept
WHERE employee.DeptNo = dept.DeptNo
```
- RA: $\pi_{\text{Emp,Mgr}}(\text{employee} \bowtie \text{dept})$
- RC: $F(\text{Emp},\text{Mgr}) =$
$\exists \text{Salary}, \text{DeptNo} : (\text{employee}(\text{Emp}, \text{Salary}, \text{DeptNo}) \wedge \text{dept}(\text{DeptNo}, \text{Mgr}))$
- **Datalog**: boss(Emp,Mgr) ← employee(Emp, Salary, DeptNo), dept(DeptNo,Mgr)