

## Smått och gott

**DB2 RUG**  
**2004-01-21--22**



Peter G Backlund  
Peter Backlund DB2-Konsult AB  
pbacklu@attglobal.net

## Innehåll



Access Path för DB2 RI

Uppdatera förälder med trigger

Identity

Skapa löpnummer med trigger

Ny SQL i DB2 LUW V8

© Peter Backlund DB2-Konsult AB

## Smått och gott

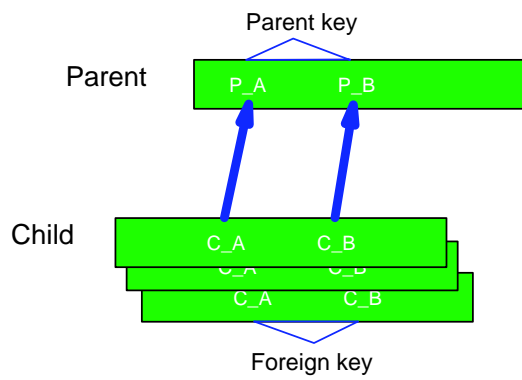
**Access Path för DB2 RI**

**DB2 RUG**  
**2004-01-21--22**



Peter G Backlund  
Peter Backlund DB2-Konsult AB  
pbacklu@attglobal.net

## Fk-1. Tabelldefinitioner



© Peter Backlund DB2-Konsult AB

## Fk-2. Pseudo SQL



Update/Delete Parent Where P\_A = :pa and P\_B = :pb

-----

Select 0 From Child  
Where C\_A = :pa and C\_B = :pb  
Fetch first 1 row only

Update Child Set C\_A = NULL, C\_B = NULL  
Where C\_A = :pa and C\_B = :pb

Delete from Child  
Where C\_A = :pa and C\_B = :pb

© Peter Backlund DB2-Konsult AB

## Fk-3. Where C\_A = :pa and C\_B = :pb



Index(C\_A, C\_B) ==> 2/2  
Index(C\_B, C\_A) ==> 2/2

Index(C\_A) ==> 1/1  
Index(C\_B) ==> 1/1

Index(C\_A, C\_B, C\_C) ==> 2/3  
Index(C\_B, C\_A, C\_C) ==> 2/3  
Index(C\_A, C\_C, C\_B) ==> 1/3 + screening  
Index(C\_B, C\_C, C\_A) ==> 1/3 + screening

Index(C\_C, C\_A, C\_B) ==> 0/3  
Index(C\_C, C\_B, C\_A) ==> 0/3

Tablespace scan

© Peter Backlund DB2-Konsult AB

## Fk-4. DB2 Referential Integrity



- ✓ Index(C\_A , C\_B) ==> 2/2
- ✗ Index(C\_B , C\_A) ==> 2/2
  
- ✗ Index(C\_A) ==> 1/1
- ✗ Index(C\_B) ==> 1/1
  
- ✓ Index(C\_A , C\_B , C\_C) ==> 2/3
- ✗ Index(C\_B , C\_A , C\_C) ==> 2/3
- ✗ Index(C\_A , C\_C , C\_B) ==> 1/3 + screening
- ✗ Index(C\_B , C\_C , C\_A) ==> 1/3 + screening
  
- ✗ Index(C\_C , C\_A , C\_B) ==> 0/3
- ✗ Index(C\_C , C\_B , C\_A) ==> 0/3
  
- ✓ Tablespace scan

© Peter Backlund DB2-Konsult AB

## Fk-5. Foreign Keys utan bra index



```
SELECT
strip(f.CREATOR)!!'!'!!strip(f.tbname) as child, f.COLNAME, f.COLSEQ, f.RELNAME
strip(r.refcreator)!!'!'!!strip(r.refbname) as parent
FROM SYSFOREIGNKEYS AS F
join sysrels as r
on f.creator = r.creator
and f.tbname = r.tbname
and f.colname = r.colname
WHERE f.CREATOR <> 'SYSIBM'
AND f.COLSEQ NOT IN
( (SELECT COUNT(*)
FROM SYSINDEXES AS IX
JOIN
SYSKEYS AS KY
ON IX.CREATOR = KY.IXCREATOR
AND IX.NAME = KY.IXNAME
JOIN
SYSFOREIGNKEYS AS FK
ON FK.COLNAME = KY.COLNAME
AND FK.COLSEQ = KY.COLSEQ
WHERE KY.COLSEQ <= F.COLSEQ
AND IX.TBCREATOR = F.CREATOR
AND IX.TBNAME = F.TBNAME
AND FK.CREATOR = F.CREATOR
AND FK.TBNAME = F.TBNAME
AND FK.RELNAME = F.RELNAME
GROUP BY IX.TBCREATOR,IX.TBNAME,IX.CREATOR,IX.NAME
HAVING COUNT(*) = F.COLSEQ
)
)
ORDER BY child
,RELNAME
, f.COLSEQ
WITH UR
```

© Peter Backlund DB2-Konsult AB

Smått och gott

Uppdatera förälder  
(med trigger)



DB2 RUG  
2004-01-21--22

Peter G Backlund  
Peter Backlund DB2-Konsult AB  
pbacklu@attglobal.net

## A-1. Updating Parent



- When you have a referential relationship, the parent key can not be updated
  - if children are existing
- The application solution is
  - insert new parent row
  - update foreign key in children rows
  - delete old parent row
- Handling this operation in a trigger would simplify application programming

© Peter Backlund DB2-Konsult AB

- ▶ In my first example we will take a look at a common problem when you have defined DB2 Referential Integrity. If you want to update a parent key where one or more children are existing, the DB2 rule is "restrict", i.e. not allowing the update (for DB2 LUW there is also a "no action" option, but this doesn't make any difference).
- ▶ A possible solution is to use application coding,
  - insert a new parent
  - update foreign key of child(ren)
  - delete old parent
- ▶ My customer asked me to provide a trigger solution.

## A-2. Updating Parent...



```
create table parent (pk int not null primary key, c1 int not null, c2 int not null)␣
--
create table child (pk int not null primary key
, fk int not null references parent(pk), c1 int not null , c2 int not null)␣
--
create trigger pk_u_a
after update of pk on parent
referencing new as n old as o
for each row mode db2sql
when ( exists (select 0 from child where fk = o.pk ) )
begin atomic
insert into parent(pk, c1, c2) values(new_key, n.c1, n.c2);
update child set fk = new_key where fk = o.pk;
delete from parent where pk = o.pk;
end ␣
```

© Peter Backlund DB2-Konsult AB

- ▶ We define a parent table and a child table with a foreign key referencing the parent key and insert some test data.
- ▶ As we want to modify tables, we have to create an after trigger defined for update of the parent key (pk) in the parent table and through the **when** clause specify the trigger action which is to be performed when children are existing.

## A-3. Updating Parent...



- We now try the trigger

```
Update Parent Set pk = new_key
Where pk = old_key
```
- The update will be rejected (SQLCODE -531)
  - this can be handled in a before trigger which "nullifies" the update
- The new key value has to be saved for use in the after trigger
  - there is no "scratchpad" available in triggers, this can be handled by adding a dummy column to the parent table

© Peter Backlund DB2-Konsult AB

- ▶ Next we test the trigger.
- ▶ Unfortunately the update is rejected with SQLCODE -531, SQLSTATE 23504 as DB2 tries the update before the trigger is activated; we need a before trigger to temporarily **nullify** the update.
- ▶ Another problem is the need to **propagate** the new pk value into the after trigger. As there is no "scratchpad" available in triggers, a possible solution is to use a dummy column in the parent table.

## A-4. Adding a column - problems



- Row length will be variable
  - updating any column will update new column
- Select \* into ... from ...
  - also for cursor; fetch into ...
- Insert into *table-name* values(...)

© Peter Backlund DB2-Konsult AB

## A-5. Updating Parent...



```
alter table parent add dc integer
/* dummy column for saving new key */
--
create trigger pk_u_b
no cascade before update of pk on parent
referencing new as n old as o
for each row mode db2sql
when ( exists ( select 0
                from child
                where fk = o.pk ) )
begin atomic
  set n.dc = n.pk;          /* save new key in dummy column*/
  set n.pk = o.pk;         /* reset parent key */
end
```

© Peter Backlund DB2-Konsult AB

- ▶ We alter the parent table adding a dummy column (dc) and then create a before trigger where we
  - save the new key in the dummy column
  - reset the parent key to its old value

## A-6. Updating Parent...



```
create trigger pk_u_a
after update of pk on parent
referencing new as n old as o
for each row mode db2sql
when ( exists (select 0 from child where fk = o.pk ) )
begin atomic
  insert into parent(pk, c1, c2) values(n.dc, n.c1, n.c2);

  update child set fk = n.dc where fk = o.pk;

  delete from parent where pk = o.pk;
end
```

© Peter Backlund DB2-Konsult AB

- ▶ In the after trigger we now have access to the needed values and can perform the desired actions
  - insert the new parent
  - update all children
  - delete the old parent

## A-7. Updating Parent - However...



■ The update will still be rejected in DB2 for z/OS (SQLCODE -531)



- the "set n.pk = o.pk" doesn't seem to work as intended

■ APAR PQ60664 will fix!

© Peter Backlund DB2-Konsult AB

- ▶ This solution works perfect in DB2 for Linux, UNIX, and Windows. However, in DB2 for z/OS we still get the -531 SQLCODE.
- ▶ The problem apparently is that the trigger statement set n.pk = o.pk (where we reset the key) is not executed properly or at least not before DB2 tries the update.
- ▶ This seems to be an error and has been accepted by development; the APAR number is PQ60664

## A-8. Testing APAR PQ60664



```
create table parent(pk int not null primary key);
create unique index xparent on parent(pk);
create table child(ck int references parent);
```

```
create trigger update
no cascade before update on parent
referencing new as n old as o
for each row mode db2sql
set n.pk = o.pk;
```

```
insert into parent values(10);
insert into child values(10);
update parent set pk = 20;
```

SQL -531 No Good  
SQL 0 OK

© Peter Backlund DB2-Konsult AB

## Smått och gott

### Identity

### DB2 RUG

### 2004-01-21--22



Peter G Backlund  
Peter Backlund DB2-Konsult AB  
pbacklu@attglobal.net

## Id-1. Identity



- Kom i V6; små förbättringar i V7
- Problem med Identity-kolumn i tabell
  - Ingen reset, restart, ALTER
  - Problem vid kopiering
  - RI??
  - Alltid numerisk

© Peter Backlund DB2-Konsult AB

## Id-2. Separat Identity-tabell



Create table reftab(refnr int not null  
generated always as identity)

Insert into reftab values(default)

Set :ws-refnr = identity\_val\_local()

© Peter Backlund DB2-Konsult AB

## Id-3. Rensa Identity-tabell



Insert into reftab values(default)

Delete from reftab

Commit

Set :ws-refnr = identity\_val\_local()

NULL!

Insert into reftab values(default)

Delete from reftab

Set :ws-refnr = identity\_val\_local()

-911

Insert into reftab values(default)

Set :ws-refnr = identity\_val\_local()

Delete from reftab where refnr = :ws-refnr

-911

© Peter Backlund DB2-Konsult AB

## Id-4. Identity och Savepoint



Savepoint alfa on rollback retain cursors

Insert into reftab values(default)

Set :ws-refnr = identity\_val\_local()

Rollback to savepoint alfa

© Peter Backlund DB2-Konsult AB

## Id-5. Identity\_val\_local()



Önskat resultat:

PIC X(12) VALUE '741xxxxxxxx'

Identity\_val\_local() är decimal(31,0)

Set :ws-refnr = '741' concat  
substr(digits(identity\_val\_local()),23,9)

© Peter Backlund DB2-Konsult AB

## Smått och gott

### Skapa löpnummer (med trigger)



**DB2 RUG**  
**2004-01-21--22**

Peter G Backlund  
Peter Backlund DB2-Konsult AB  
pbacklu@attglobal.net

- ▶ DB2 for z/OS V8 also has **Sequence Objects**.

## B-1. Sequencing



▶ Generating Sequence Numbers can be a problem

- Identity
  - difficult to modify the values
  - can generate holes
- Sequence Objects
  - DB2 LUW only - **new in DB2 z/OS V8**
  - can generate holes
- "Next Number" Table
  - application coding
  - bottleneck if no holes

© Peter Backlund DB2-Konsult AB

## B-2. Sequencing with Triggers



- Using a "Next Number" Table (**nextsn**)
- Avoiding application coding
- No holes - may be a bottleneck!
- Before trigger - gets sequence number from **nextsn** table
- After trigger - updates sequence number in **nextsn** table

© Peter Backlund DB2-Konsult AB

- ▶ One customer suggested a solution using a "Next Number" table and triggers, thus avoiding application coding.
- ▶ A before insert trigger could get the sequence number and then an after insert trigger could be used to update the sequence number table.
- ▶ This solution would also guarantee that there would be no holes in the sequence number series. This may be important for some applications, but may also produce a bottleneck, as transactions would be serialized on the insert.

## B-3. Sequencing Example...



```
create table stryker (pk integer not null, pn integer)
create table nextsn (sk integer not null, sn integer not null)
```

```
create trigger sb
no cascade before insert on stryker
referencing new as n
for each row mode db2sql
set n.pn = (select sn + 1
           from nextsn
           where sk = n.pk)
```



```
create trigger sa
after insert on stryker
referencing new as n
for each row mode db2sql
update nextsn set sn = sn+1
where sk = n.pk
```

pk = category  
pn = sequence number

sk = category  
sn = next sequence number

© Peter Backlund DB2-Konsult AB

- ▶ This code example shows the two triggers; the sb (before) picks up next sequence number and the sa (after) updates the sequence number table.
- ▶ This works perfect in DB2 for Linux, UNIX, and Windows, but **not in DB2 for z/OS and OS/390!**

## B-4. Restriction in DB2 for z/OS



■ No **subselect** in the trigger body  
(only in the when clause)

▶ The solution:

- Stored Procedures  
(not in DB2 LUW)
- User Defined Functions  
(SQL or External)

© Peter Backlund DB2-Konsult AB

- ▶ The limitation in DB2 for z/OS and OS/390 is that subselects are not allowed in the body of a trigger - they can only be used in the WHEN clause (which is used to decide if the trigger action should be effectuated).
- ▶ Possible solutions are to use User Defined Functions or Stored Procedures. The latter can not be used in DB2 for Linux, UNIX, and Windows, as the CALL statement is not allowed in triggers.
- ▶ For UDFs the possibilities are to use SQL or External Functions.

## B-5. Sequencing Example...



```
create function f_nextsn (pk integer)
returns integer
language sql
  return select sn + 1
         from nextsn
         where sk = pk
```



```
create trigger sb
no cascade before insert on stryker
referencing new as n
for each row mode db2sql
set n.pn = f_nextsn(n.pk)
```

```
create trigger sa
after insert on stryker
referencing new as n
for each row mode db2sql
  update nextsn set sn = sn+1
  where sk = n.pk
```

© Peter Backlund DB2-Konsult AB

- ▶ So, we create an SQL UDF (also named nextsn) which returns next sequence number for a given key.
- ▶ The before trigger uses this function to retrieve next sequence number and the after trigger does the update.
- ▶ This works perfect in DB2 for Linux, UNIX, and Windows, but **not in DB2 for z/OS and OS/390!**

## B-6. Restriction in DB2 for z/OS



■ User Defined Functions - language SQL (new in V7)  
can not have **SQL** in the return clause

▶ The solution:

- External User Defined Function  
written in COBOL, PL/I, C,...
- Possible performance penalty  
~ separate external module  
~ separate package

→ And - External UDFs in DB2 LUW can not contain SQL!

→ In DB2 LUW V8.1 an External UDF can have SELECT

© Peter Backlund DB2-Konsult AB

- ▶ New in DB2 for Linux, UNIX, and Windows V8 is that SQL can be used in external UDFs - as long as it is SELECT.

## B-7. Sequencing - Another Solution



```
create trigger sa
after insert on stryker
referencing new as n
for each row mode db2sql
begin atomic
  update nextsn set sn = sn+1
  where sk = n.pk;
  update stryker set pn =
    (select sn
     from nextsn
     where sk = n.pk)
  where pk = n.pk;
end
```

- ▶ But - must have valid (dummy) value for sn at insert
- could use a before trigger

© Peter Backlund DB2-Konsult AB

- ▶ A completely different solution would be to use an after insert trigger to do both updating of the sequence number table and assigning of this number to the table.
- ▶ The only problem is that the sequence number which is used for the original insert has to be valid. This could be some dummy number or NULL and could be assigned using a before trigger.

## Smått och gott

### Ny SQL i DB2 LUW V8

**DB2 RUG**  
**2004-01-21--22**



Peter G Backlund  
Peter Backlund DB2-Konsult AB  
pbacklu@attglobal.net

## Ny SQL i DB2 LUW V8



Version 8.1

- INSERT i VIEW med UNION ALL
- INSTEAD OF triggers
- SQL i extern UDF (fast bara SELECT!)

Fixpack 2

- Sampling
- MERGE

Fixpack 4

- SELECT med INSERT, UPDATE, DELETE

© Peter Backlund DB2-Konsult AB

## a-1 View med Union All



En View med UNION ALL kan användas för att "simulera" ett partitionerat tablespace

### Fördelar

- kan enkelt lägga till och ta bort "partitioner"
- tabellerna kan ha olika clustering index
- ingen NPI

### Nackdelar

- ingen NPI
- Read-Only - **åtminstone i DB2 for z/OS**

© Peter Backlund DB2-Konsult AB

## a-2 View med Union All - DB2 LUW



I DB2 for Linux, UNIX, Windows tillåts för en View med UNION ALL (såvida det inte är en Read-Only View)

### I V7.2

- Update
- Delete

### I V8.1

- Insert

V8.1

© Peter Backlund DB2-Konsult AB

## a-3 Tabeller och view



```
Create table empa
(empno char(2) not null
, lastname char(4) not null
, workdept char(2) not null); /* workdept börjar med A */
```

```
Create table empb like empa; /* workdept börjar med B */
Create table empc like empa; /* workdept börjar med C */
```

```
Create view empall as
select * from empa union all
select * from empb union all
select * from empc;
```

© Peter Backlund DB2-Konsult AB

## a-4 Delete, Update mot View



Delete from empall where empno = ?;

Delete from empall where workdept like ?;

Update empall set lastname = ? where empno = ?;

Update empall set workdept = 'A1'

where empno = '30';

/\* nuvarande workdept är C1 \*/

/\*???? Raden ligger kvar i empc ????\*/

© Peter Backlund DB2-Konsult AB

## a-5 Begränsa Update mot View



Create view empall as

select \* from emp\_a where workdept like 'A%'

union all

select \* from emp\_b where workdept like 'B%'

union all

select \* from emp\_c where workdept like 'C%'

with check option;

Update empall set workdept = 'A1'

where empno = '30';

/\* resulterar i SQLCODE -161 \*/

© Peter Backlund DB2-Konsult AB

## a-6 Insert mot View



Insert into empall (empno, lastname, workdept)

values ('11', 'Anna', 'A1');

SQL20154N The requested insert operation into view *empall* is not allowed because no target table can be determined for a row. Reason code = 2.

1. does not satisfy the check constraint of any underlying base table
2. satisfies all the check constraints for more than one underlying base table.

© Peter Backlund DB2-Konsult AB

## a-7 Insert mot View - lösning



```
Alter table emp_a add check(workdept like 'A%');
Alter table emp_b add check(workdept like 'B%');
Alter table emp_c add check(workdept like 'C%');
```

```
Create view emp_all as
select * from emp_a union all
select * from emp_b union all
select * from emp_c;
```

Nu går det att göra Insert!

Begränsar också Update!

© Peter Backlund DB2-Konsult AB

## b-1 INSTEAD OF triggers



■ En read-only view kan normalt bara användas för läsning (SELECT)

V8.1

■ "Instead of trigger" möjliggör

- INSERT
- UPDATE
- DELETE

mot en sådan view

© Peter Backlund DB2-Konsult AB

## b-2 Tabeller och View



```
Create Table Dept
( Deptno char(2)
, Deptname char(5));
```

Deptno	Deptname
A1	Alfa
B1	Beta
C1	Gamma
D1	Delta

```
Create Table Emp
( Empno char(2)
, Lastname char(4)
, Workdept char(2));
```

Empno	Lastname	Workdept
10	Alf	A1
20	Bea	C1
30	Carl	C1
50	Eva	D1
60	Fred	E1
70	Gus	A1

```
Create view Empv as
Select Empno, Lastname, Deptname
From Dept join Emp
on Workdept = Deptno;
```

Empno	Lastname	Deptname
10	Alf	Alfa
20	Bea	Gamma
30	Carl	Gamma
50	Eva	Delta
70	Gus	Alfa

© Peter Backlund DB2-Konsult AB

### b-3 Instead of DELETE



Create trigger empv\_delete  
instead of delete on empv  
referencing old as oldemp  
for each row mode db2sql  
delete from emp as e  
where e.empno = oldemp.empno

### b-4 Instead of DELETE - exempel



delete from empv  
where empno = '10';

Empno	Lastname	Workdept
10	Alf	A1
20	Bea	C1
30	Carl	C1
50	Eva	D1
60	Fred	E1
70	Gus	A1

delete from empv  
where deptname = 'Gamma';

Empno	Lastname	Workdept
20	Bea	C1
30	Carl	C1
50	Eva	D1
60	Fred	E1
70	Gus	A1

Empno	Lastname	Workdept
50	Eva	D1
60	Fred	E1
70	Gus	A1

### b-5 Instead of INSERT



Create trigger empv\_insert  
instead of insert on empv  
referencing new as newemp  
for each row mode db2sql  
insert into emp (empno,lastname, workdept)  
values(empno, lastname,  
coalesce((select deptno from dept as d  
where d.deptname = newemp.deptname),  
raise\_error('70001', 'Unknown Dept')))

## b-6 Instead of INSERT - exempel



Empno	Lastname	Workdept
10	Alf	A1
20	Bea	C1
30	Carl	C1
50	Eva	D1
60	Fred	E1
70	Gus	A1

```
Insert into rug.empv  
values('11','Anna','Alfa');
```

```
Insert into rug.empv  
values('12','Eva','SEB');
```

SQL0438N Application raised  
error with diagnostic text  
"Unknown Dept"  
SQLSTATE 70001

Empno	Lastname	Workdept
10	Alf	A1
11	Anna	A1
20	Bea	C1
30	Carl	C1
50	Eva	D1
60	Fred	E1
70	Gus	A1

© Peter Backlund DB2-Konsult AB

## b-7 Instead of UPDATE



```
create trigger empv_update  
instead of update on empv  
referencing new as n old as o  
for each row mode db2sql  
update emp as e  
set (empno, lastname, workdept)  
= (n.empno, n.lastname,  
coalesce((select deptno from dept as d  
where d.deptname = n.deptname),  
raise_error ('70001', 'Unknown Dept')))  
where n.empno = e.empno
```

© Peter Backlund DB2-Konsult AB

## b-8 Instead of UPDATE - kul grej



```
begin atomic  
values(case when n.empno = o.empno then 0  
else raise_error('70002', 'Must not change empno')  
end);  
update emp as e  
set (empno, lastname, workdept)  
= (n.empno, n.lastname,  
coalesce((select deptno from dept as d  
where d.deptname = n.deptname),  
raise_error ('70001', 'Unknown Dept')))  
where n.empno = e.empno;  
end
```

© Peter Backlund DB2-Konsult AB

## c-1 Sampling



Select ...  
from ...

Fixpack  
2

tablesample bernoulli (p) -- 0<p<=100  
system

[repeatable (n)]

where ...

## c-2 Sampling - testtabell



Select count(\*) from t\_emp;

100,000 rader  
200,000  
...  
1,000,000 rader

Select count(\*) from t\_emp  
where workdept like 'A%';

Empno	Workdept	Newcol
Integer	Char(3)	Char(13)

Alter table t\_emp add  
newcol char(13)  
not null with default;



## c-3 Materialized Query Table



Create table mqt\_emp as  
(select workdept as avd, count(\*) as antal  
from t\_emp  
group by workdept)  
data initially deferred refresh immediate;

avd	antal
char(3)	integer

Set integrity for mqt\_emp immediate  
checked;

====> 33 rader

### c-4 Slutlig testtabell

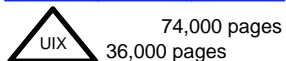


Tabell t\_emp - 10,000,000 rader

Select count(\*)  
from t\_emp  
where newcol like 'A%';

Empno	Workdept	Newcol
Integer	Char(3)	Char(13)
10M	33	10M

Select max(newcol)  
from t\_emp;



### c-5 Test med tablesample



Select count(\*)\*100/p from t\_emp  
tablesample system (p)  
where newcol like 'A%';

P	Bernoulli	System
-----	13,00 - 10M	13,00 - 10M
100	13,65 - 10.00	12,97 - 10.00
10	13,75 - 10.01	1,40 - 10.03
1	13,75 - 10.03	0,14 - 9.83
0.1	14,03 - 10.03	0,02 - 10.32

### c-6 Test med tablesample



Select max(newcol) from t\_emp  
tablesample system (p);

P	Bernoulli	System
-----	13,43 - 10000000	13,43 - 10000000
100	13,55 - 10000000	13,50 - 10000000
10	12,79 - 9999914	1,77 - 9999922
1	13,04 - 9996519	0,13 - 9993759
0.1	12,68 - 9993695	0,01 - 9977418

## d-1 Merge



MERGE är en ny SQL-sats

Fixpack 2

"Kombinerar" Insert och Update

Data i en transaktionstabell "mergas"  
in i en mastertabell baserat på nyckelvärde:

- om värdet finns: Update
- om värdet inte finns: Insert

© Peter Backlund DB2-Konsult AB

## d-2 MERGE med trans-tabell



```
MERGE INTO emp AS t
USING (select * from trans) as s
ON t.empno = s.empno
WHEN MATCHED THEN
UPDATE SET
    salary = s.salary, lastname = s.lastname
, sex = s.sex, workdept = s.workdept
, hiredate = s.hiredate
WHEN NOT MATCHED THEN
INSERT
VALUES (s.empno, s.lastname, s.sex
, s.workdept, s.salary, s.hiredate);
```

© Peter Backlund DB2-Konsult AB

## d-3 MERGE med trans-data



```
MERGE INTO emp AS t
USING TABLE (VALUES
('20', null, null, null, 35000, null)
,('90','Carl', 'M','C1', 33000, '1963-03-03'))
s(empno, lastname, sex, workdept, salary, hiredate)
ON t.empno = s.empno
WHEN MATCHED THEN
UPDATE SET
    salary = s.salary
WHEN NOT MATCHED THEN
INSERT
VALUES (s.empno, s.lastname, s.sex
, s.workdept, s.salary, s.hiredate);
```

© Peter Backlund DB2-Konsult AB

## d-4 MERGE i Trigger



```
Create trigger ainsdet
after insert on detail
referencing new_table as n
for each statement mode db2sql
MERGE INTO summary
USING (select did, sum(amount) s_amount from n
      group by did) d
ON (did = sid)
WHEN MATCHED THEN
  UPDATE SET total = total + s_amount
WHEN NOT MATCHED THEN
  INSERT VALUES (did,s_amount);
```

© Peter Backlund DB2-Konsult AB

## e-1 Select med Insert, Update, Delete



```
      old
Select ... from new table
      final
(
  insert ...
  update ...
  delete ...
);
```

Fixpack 4

© Peter Backlund DB2-Konsult AB

## e-2 Old, New, and Final Table



	Insert	Update	Delete
Old table	-	Y	Y
New table	Y	Y	-
Final table	Y	Y	-

Old - data before update or delete  
New - data before update, insert, and triggers  
Final - data after update and insert

© Peter Backlund DB2-Konsult AB

### e-3 Select med Insert - exempel



```
Create table master  
( empno int not null generated always as identity  
 , name char(10) not null  
 , tidpunkt time not null with default  
 , salary int not null);
```

```
select empno, name, tidpunkt from final table(  
insert into master (name, salary) values  
( 'Backlund',10),  
( 'Gademan',20));
```

<i>empno</i>	<i>name</i>	<i>tidpunkt</i>
10	Backlund	17:28:28
11	Gademan	17:28:28

© Peter Backlund DB2-Konsult AB

### e-4 Insert från Select med Insert



```
create table audit  
(empno int not null  
 , name char(10) not null  
 , tidpunkt time not null);
```

```
insert into audit  
select empno, name, tidpunkt from final table(  
insert into master (name, salary) values  
( 'Backlund',10));
```

SQLCODE -20165  
SQLSTATE 428FL

© Peter Backlund DB2-Konsult AB

