Firebird, ¿mejor en Windows o Linux?

Vamos a realizar las pruebas con la siguiente base de datos

```
SET SQL DIALECT 3;
CREATE DATABASE 'C:\testeo\testeo.fdb'
USER 'sysdba' PASSWORD 'el de siempre'
PAGE_SIZE 4096
DEFAULT CHARACTER SET WIN1251;

CREATE GENERATOR SACARID ;

SET GENERATOR SACARID TO 0;

CREATE TABLE TABLA1 (
ID INTEGER NOT NULL,
CAMPO1 VARCHAR (80) CHARACTER SET WIN1251 NOT NULL COLLATE WIN1251,
CAMPO2 VARCHAR (80) CHARACTER SET WIN1251 NOT NULL COLLATE WIN1251
);

CREATE TABLE TABLA2 (
ID INTEGER NOT NULL,
VALOR1 VARCHAR (80) CHARACTER SET WIN1251 NOT NULL COLLATE WIN1251
);

ALTER TABLE TABLA1 ADD CONSTRAINT PK_TABLA1 PRIMARY KEY (ID);
ALTER TABLE TABLA2 ADD CONSTRAINT PK_TABLA2 PRIMARY KEY (ID);
ALTER TABLE TABLA2 ADD CONSTRAINT FK_TABLA2 FOREIGN KEY (ID)
REFERENCES TABLA1 (ID) ON DELETE CASCADE ON UPDATE CASCADE;

CREATE INDEX FK_TABLA2 ON TABLA2 (ID);
CREATE UNIQUE INDEX IDX_TABLA1 ON TABLA1 (CAMPO1);
CREATE UNIQUE INDEX IDX_TABLA11 ON TABLA1 (CAMPO2);
CREATE UNIQUE INDEX IDX_TABLA2 ON TABLA2 (VALOR1);
CREATE UNIQUE INDEX PK_TABLA1 ON TABLA1 (ID);
CREATE UNIQUE INDEX PK_TABLA2 ON TABLA2 (ID);

SET TERM ^ ;

CREATE PROCEDURE DAMETIEMPOS
RETURNS (
TIEMPOINICIO TIMESTAMP,
TIEMPOFINAL TIMESTAMP,
TIEMPO INTEGER)
AS
BEGIN
EXIT;
END
^

CREATE PROCEDURE METEENTABLA2 (
VALORDEID INTEGER)
AS
BEGIN
EXIT;
END
^

ALTER PROCEDURE DAMETIEMPOS
RETURNS (
TIEMPOINICIO TIMESTAMP,
TIEMPOFINAL TIMESTAMP,
TIEMPO INTEGER)
AS
DECLARE VARIABLE I INTEGER;
DECLARE VARIABLE VALORCAMPO1 VARCHAR(80);
```

```
DECLARE VARIABLE VALORCAMPO2 VARCHAR(80);
DECLARE VARIABLE tSegundosInicio INTEGER;
DECLARE VARIABLE tSegundosFinal INTEGER;
begin
TiempoInicio = 'Now';
tSegundosInicio =
(EXTRACT(Hour from TiempoInicio) * 3600 ) +
(EXTRACT(Minute from TiempoInicio) * 60 ) +
EXTRACT(Second from TiempoInicio) ;

I = 0;
WHILE (I <= 5000000) DO
BEGIN
VALORCAMPO1 = 'Valor Campo1: ' || CAST(I as varchar(80));
VALORCAMPO2 = 'Valor Campo2: ' || CAST(I as varchar(80));
INSERT INTO  TABLA1  ( tabla1.id ,  tabla1.campo1, tabla1.campo2 )
values(:I, :VALORCAMPO1, :VALORCAMPO2 );
I = I + 1;
END

TiempoFinal = 'Now';
tSegundosFinal =
(EXTRACT(Hour from TiempoFinal) * 3600 ) +
(EXTRACT(Minute from TiempoFinal) * 60 ) +
EXTRACT(Second from TiempoFinal) ;
TIEMPO = (tSegundosFinal - tSegundosInicio);

Suspend;
end
^

ALTER PROCEDURE METEENTABLA2 (
VALORDEID INTEGER)
AS
DECLARE VARIABLE IDENSTRING VARCHAR(20);
begin
IDENSTRING = 'Valor de Id: ' || CAST(:VALORDEID as varchar(80));
INSERT INTO tabla2 ( tabla2 . id , tabla2 . valor1 ) values(:VALORDEID,
:IDENSTRING );

suspend;
end
^
SET TERM ; ^

SET TERM ^ ;

CREATE TRIGGER INSERTAENTABLA2 FOR TABLA1 ACTIVE
AFTER INSERT POSITION 0
AS
begin
execute procedure meteentabla2(new. id );
end
^

CREATE TRIGGER TABLA1OPTENERID FOR TABLA1 ACTIVE
BEFORE INSERT POSITION 0
AS
BEGIN
if (new. id is null) then
new. id = gen_id( sacarid , 1);
END
^
SET TERM ; ^
```

Se hace notar que pa meterle más cañita será un alta masiva de 5.000.000 filas, tal y como se refleja

en el Store Procedure.

Pues bien, pasando ya de historias, y dado que el mismo procedure nos devuelve el tiempo de proceso, y lanzado tanto en Win2000 y Linux mediante la propia consola isql.

```
SELECT TIEMPO FROM DAMETIEMPOS;
```

Tanto en async como con sync los tiempos obtenidos han sido:

Sistema Operativo	Tipo (cache)	Tiempo (segs.)
Windows 2000 profesional	Async	937
Windows 2000 profesional	Sync	1085
Linux Mandrake 10 (kernel 2.6)	Async	461
Linux Mandrake 10 (kernel 2.6)	Sync	523

Nota: a su vez lo he lanzado también desde la WmWare y en Async el timpo ha sido de 626.

Julio Nogueira Fandiño
CombatF2D
07 de octubre de 2004