

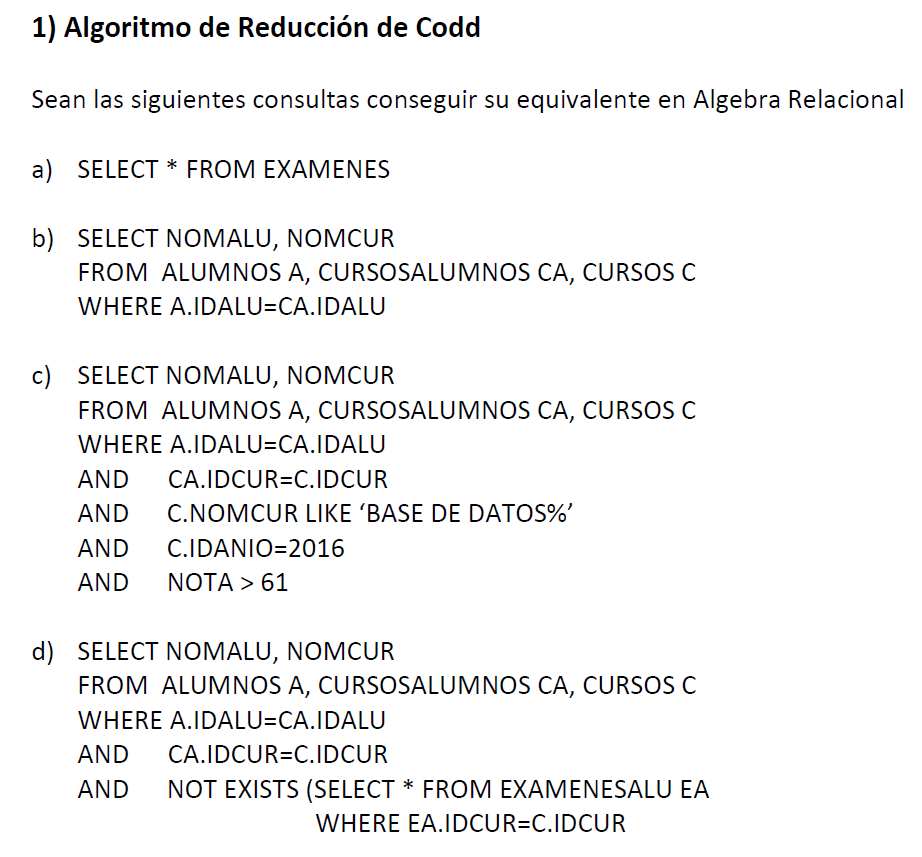
**Práctico de optimización**

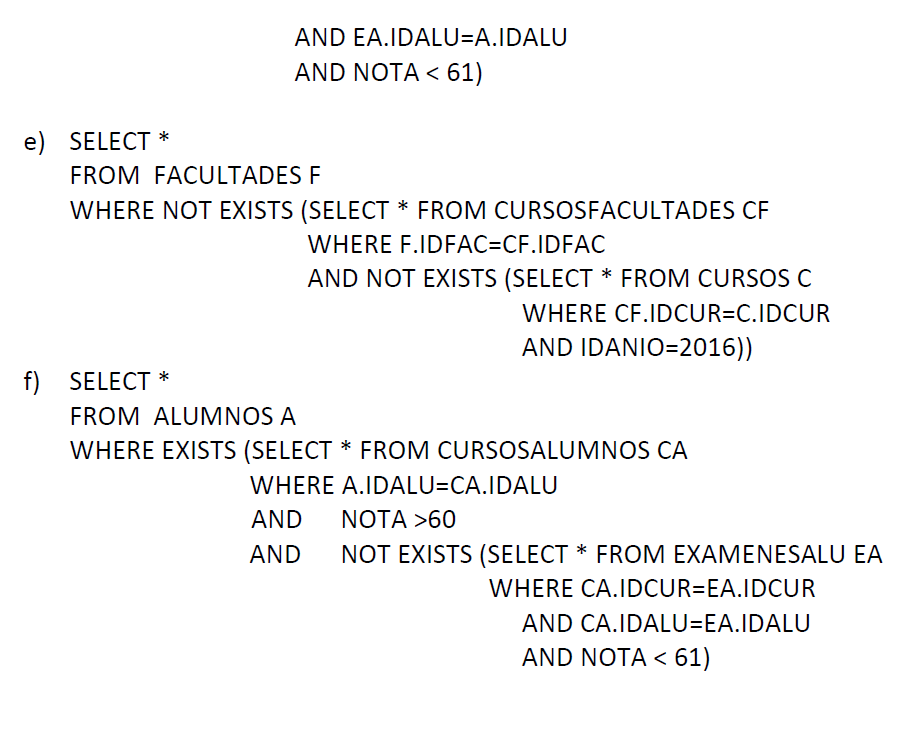
**Asignatura**: Base de Datos II

**Alumno:**

Juan Pérez 4.673.899-0

Montevideo, 05 de junio de 2023





A) σ(E)

* π∩

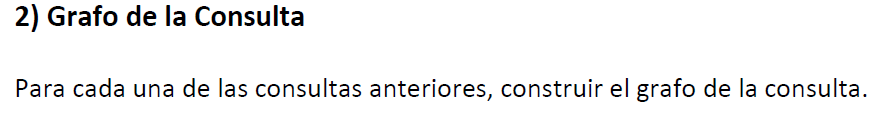
B) π NOMALU, NOMCUR (σ A.IDALU=CA.IDALU(A x C x CA))

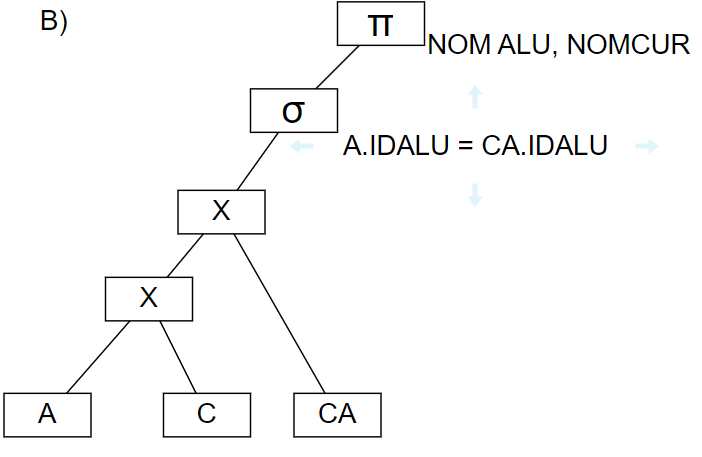
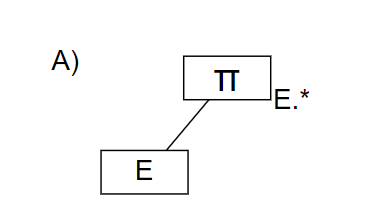
C) π NOMALU, NOMCUR (σ A.IDALU = CA.IDALU AND CA.IDCUR=C.IDCUR AND C.NOMCUR LIKE "BASE DE DATOS%"AND C.IDANIO=2016 AND NOTA > 61 (A x C x CA))

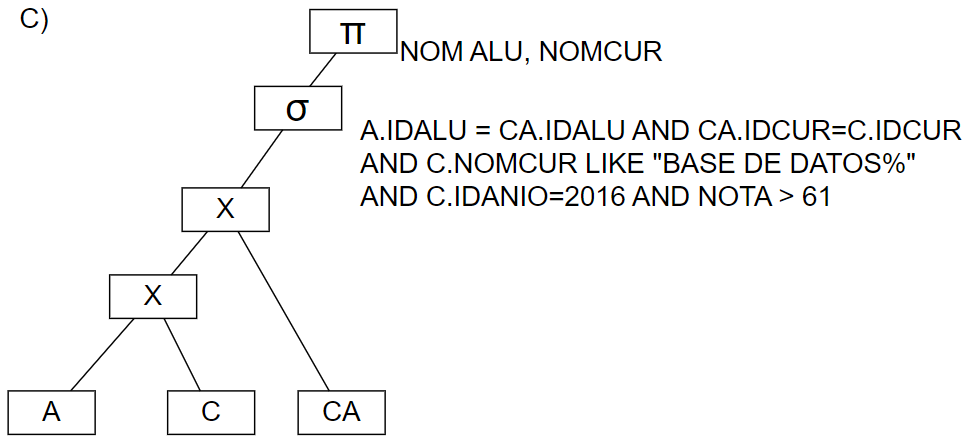
D) π NOMALU, NOMCUR ( σ A.IDALU = CA.IDALU AND CA.IDCUR=C.IDCUR (A x C x CA) – (σ EA.IDCUR=C.IDCUR AND EA.IDALU=A.IDALU AND NOTA<61 (A x C x CA x EA)))

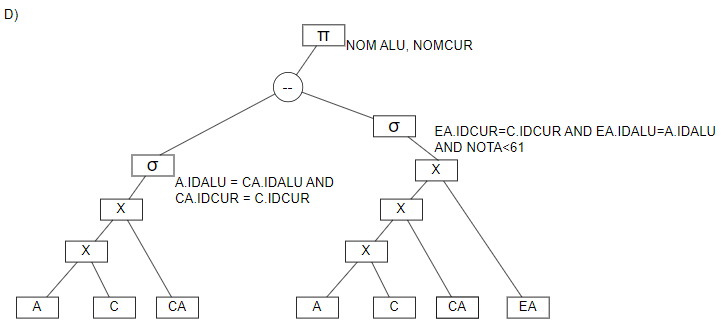
E) π F.\* (F % σ CF.IDCUR=C.IDCUR AND IDANIO = 2016( C x CF) % F)

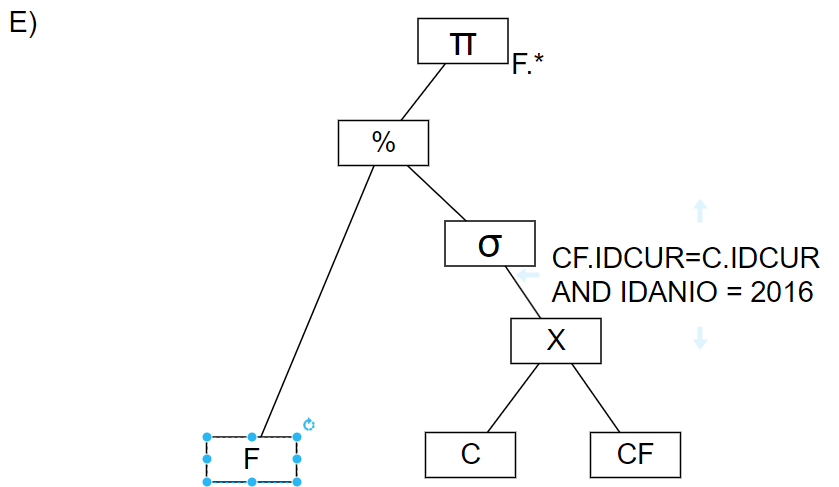
F) π A.\* (A ∩ (σ A.IDALU=CA.IDALU AND NOTA >60 (A x CA) – (σ EA.IDCUR=CA.IDCUR AND EA.IDALU=CA.IDALU AND NOTA<61 (EA x CA))))

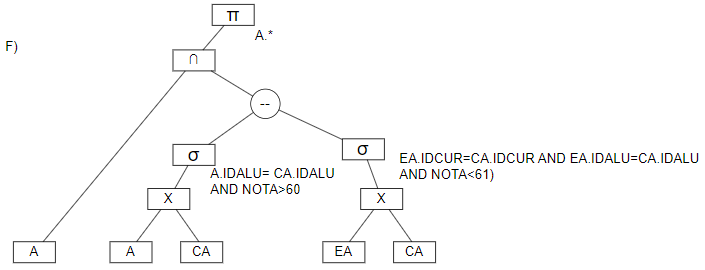


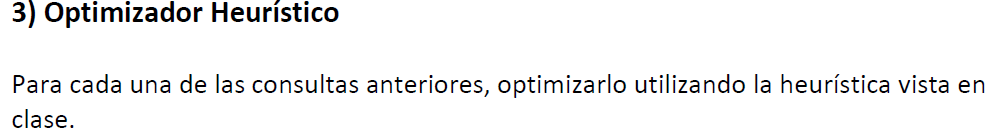


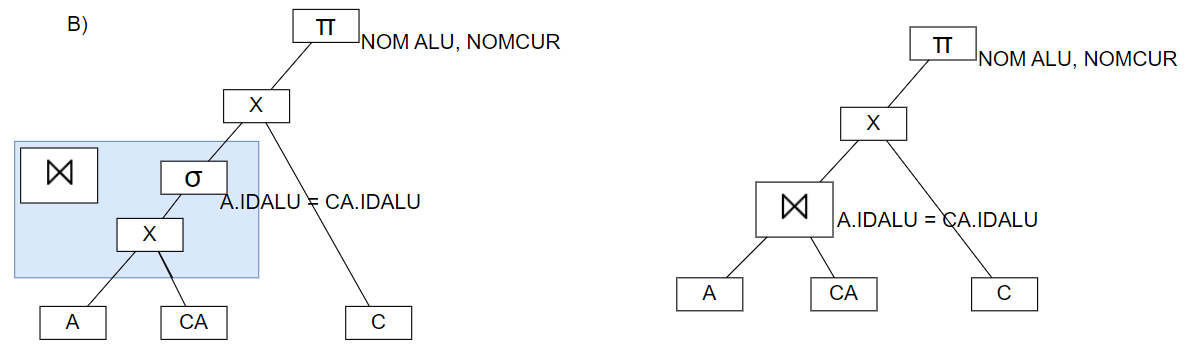


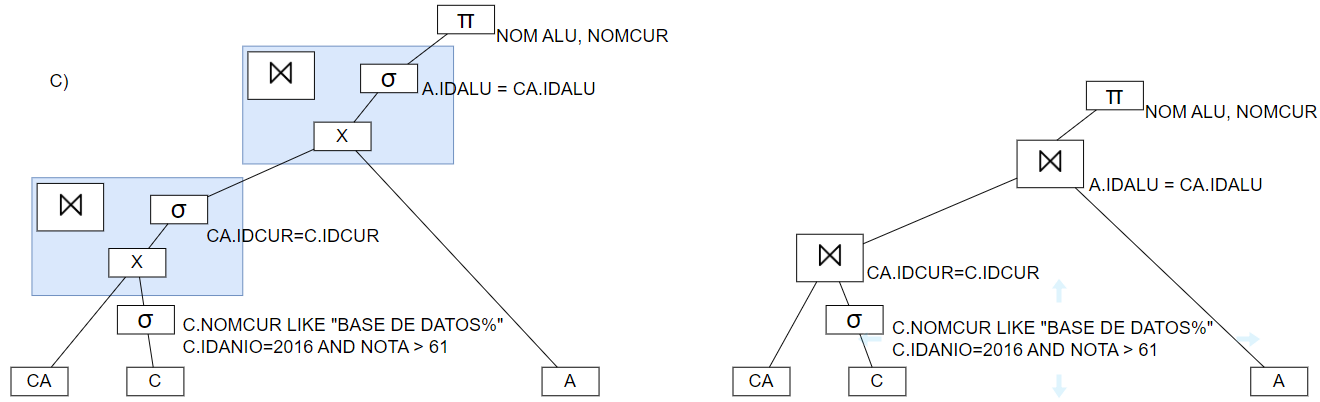


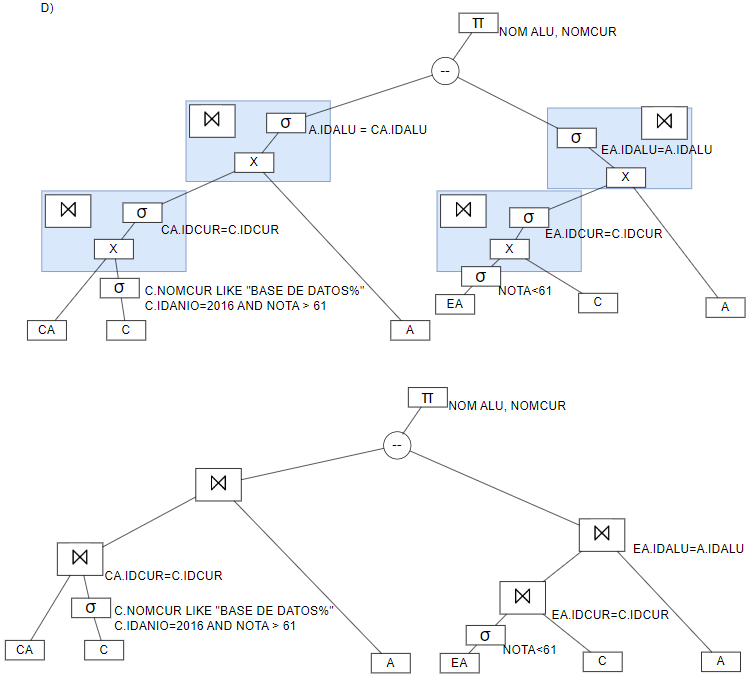


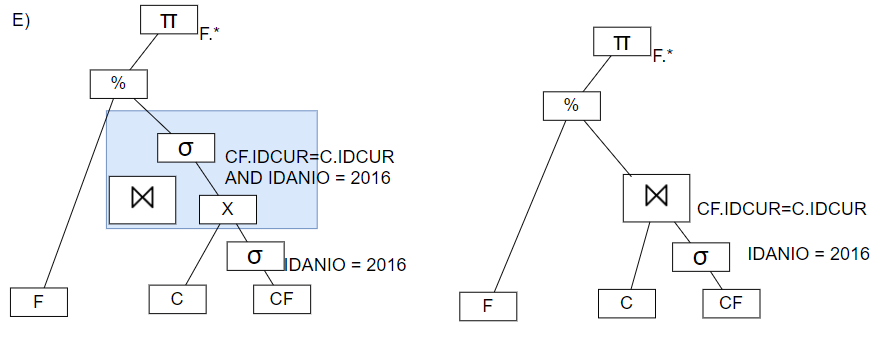


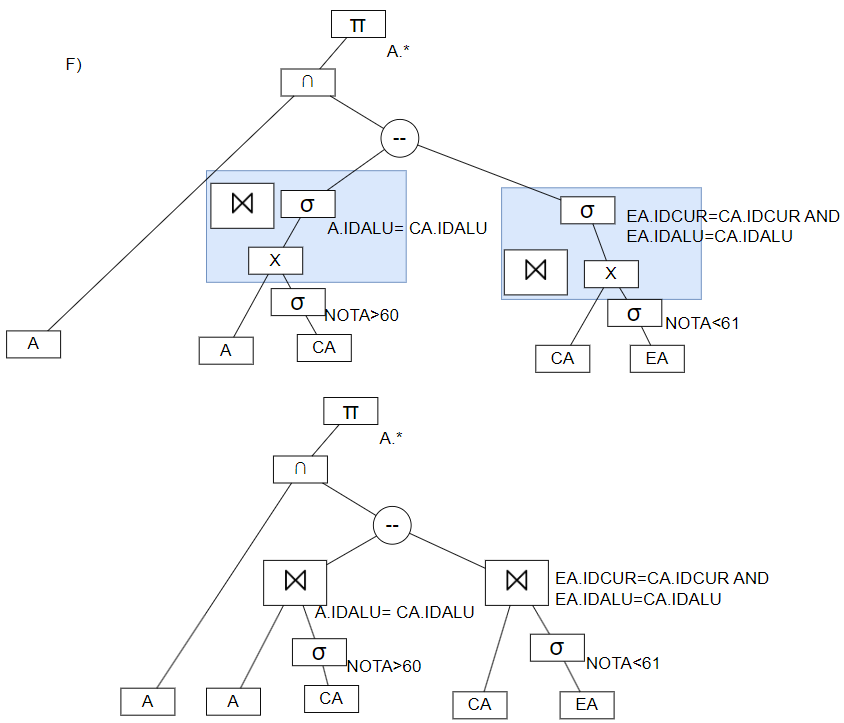


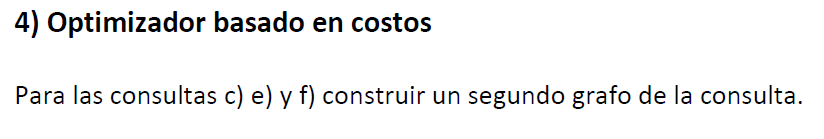


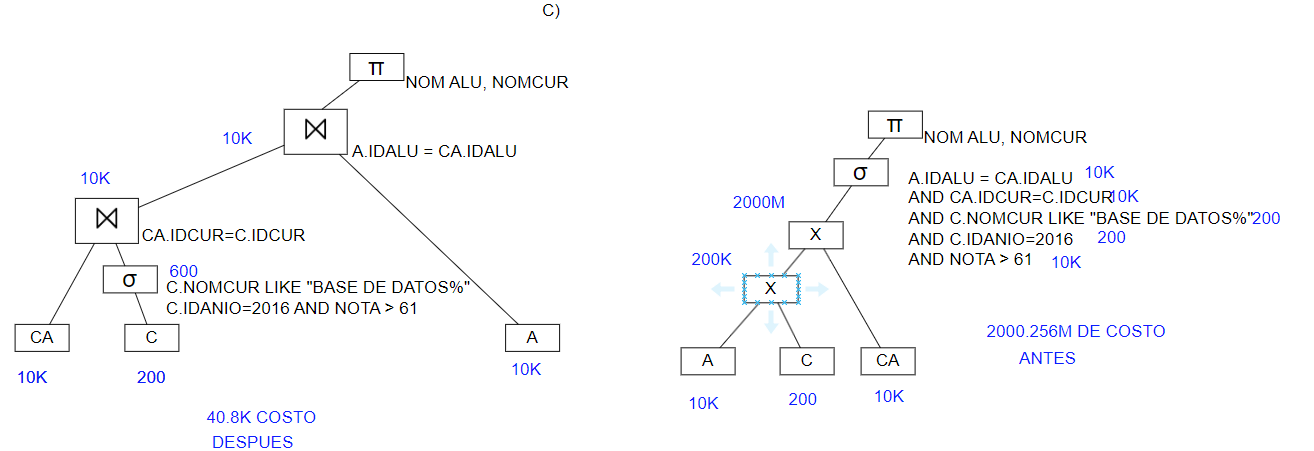


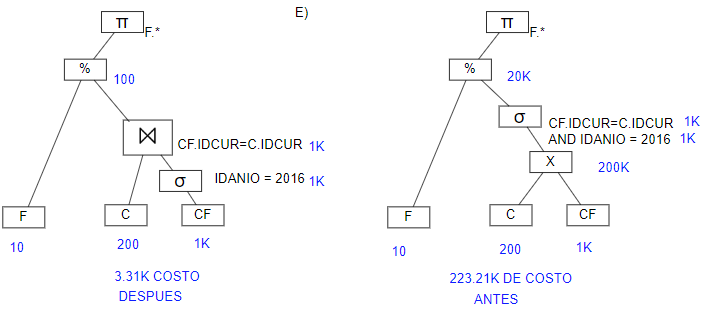


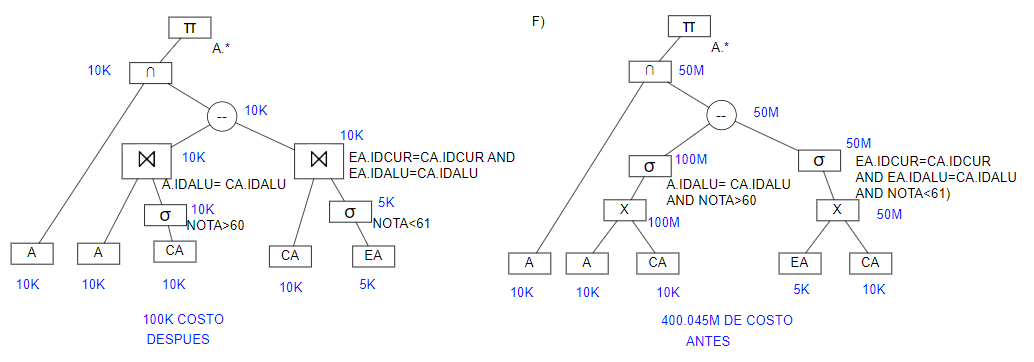


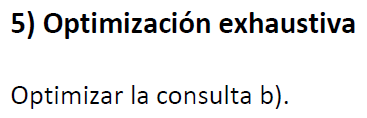


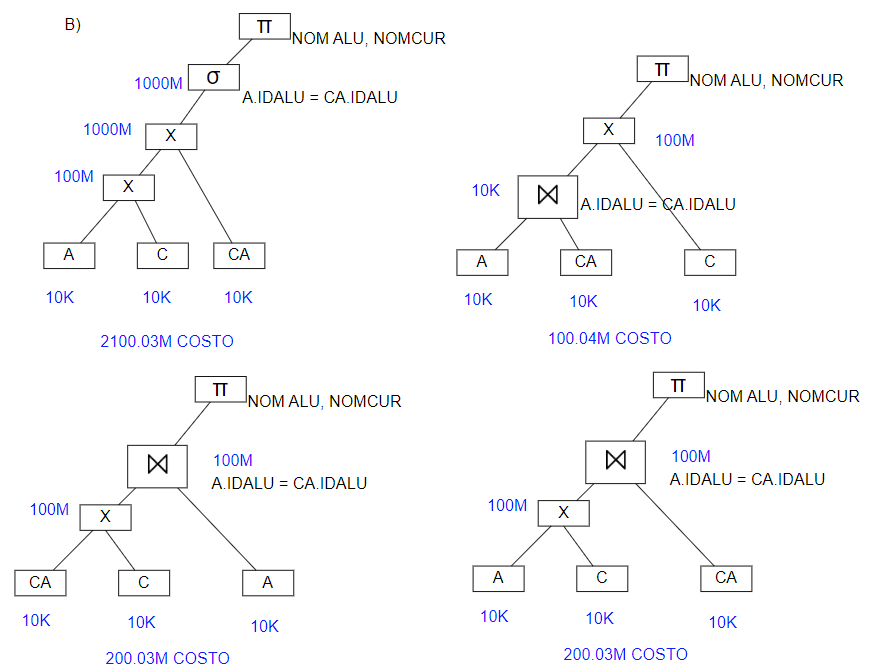


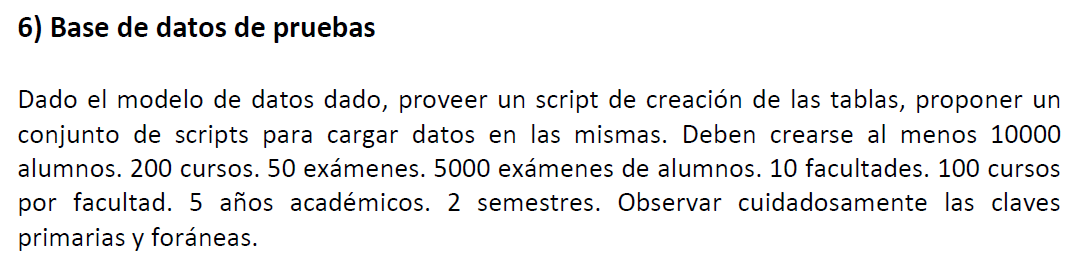












CREATE TABLE IF NOT EXISTS optimizacion.alumnos

(

"idAlu" bigint NOT NULL,

nomalu character(30) COLLATE pg\_catalog."default",

apealu character(30) COLLATE pg\_catalog."default",

docid character(30) COLLATE pg\_catalog."default",

CONSTRAINT alumnos\_pkey PRIMARY KEY ("idAlu")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.alumnos

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.semestres

(

"idSem" bigint NOT NULL,

nomsem character(30) COLLATE pg\_catalog."default",

abrsem character(30) COLLATE pg\_catalog."default",

CONSTRAINT semestres\_pkey PRIMARY KEY ("idSem")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.semestres

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.anioacademico

(

"idAnio" bigint NOT NULL,

nomAnio character(30) COLLATE pg\_catalog."default",

"idSem" bigint,

CONSTRAINT "anioAcademico\_pkey" PRIMARY KEY ("idAnio"),

CONSTRAINT "anioAcademico\_idSem\_fkey" FOREIGN KEY ("idSem")

REFERENCES optimizacion.semestres ("idSem") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.anioacademico

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.facultades

(

idFac bigint NOT NULL,

nomfac character COLLATE pg\_catalog."default",

CONSTRAINT facultades\_pkey PRIMARY KEY ("idFac")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.facultades

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.examenes

(

"idExm" bigint NOT NULL,

fchexm date,

hraexm time without time zone,

CONSTRAINT examenes\_pkey PRIMARY KEY ("idExm")

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.examenes

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.cursosfacultades

(

"idFac" bigint,

"idCur" bigint,

CONSTRAINT "cursosFacultades\_idCur\_fkey" FOREIGN KEY ("idCur")

REFERENCES optimizacion.cursos ("idCur") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT "cursosFacultades\_idFac\_fkey" FOREIGN KEY ("idFac")

REFERENCES optimizacion.facultades ("idFac") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.cursosfacultades

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.cursos

(

"idCur" bigint NOT NULL,

"idAnio" bigint,

nomcur character(30) COLLATE pg\_catalog."default",

abrcur character(10) COLLATE pg\_catalog."default",

CONSTRAINT cursos\_pkey PRIMARY KEY ("idCur"),

CONSTRAINT "cursos\_idAnio\_fkey" FOREIGN KEY ("idAnio")

REFERENCES optimizacion.anioacademico ("idAnio") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.cursos

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.cursosalumnos

(

"idAlu" bigint,

"idCur" bigint,

nota integer,

CONSTRAINT "cursosalumnos\_idAlu\_idCur\_key" UNIQUE ("idAlu", "idCur"),

CONSTRAINT "cursosAlumnos\_idAlu\_fkey" FOREIGN KEY ("idAlu")

REFERENCES optimizacion.alumnos ("idAlu") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT "cursosAlumnos\_idCur\_fkey" FOREIGN KEY ("idCur")

REFERENCES optimizacion.cursos ("idCur") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.cursosalumnos

OWNER to postgres;

CREATE TABLE IF NOT EXISTS optimizacion.examenesalu

(

"idCur" bigint,

"idAlu" bigint,

"idExm" bigint,

nota integer,

CONSTRAINT "examenesalu\_idCur\_idAlu\_fkey" FOREIGN KEY ("idAlu", "idCur")

REFERENCES optimizacion.cursosalumnos ("idAlu", "idCur") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT "examenesalu\_idExm\_fkey" FOREIGN KEY ("idExm")

REFERENCES optimizacion.examenes ("idExm") MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE IF EXISTS optimizacion.examenesalu

OWNER to postgres;

**Consultas para la inserción de datos**

<https://github.com/jumpert/PracticoOptimizacionBD2/blob/main/DatosPracticoOptimizacion.xlsx>

INSERT INTO optimizacion.alumnos VALUES (92072, 'Levi', 'Murray', '2979882');

INSERT INTO optimizacion.examenes VALUES (1001, ' 2011-01-01', ' 08:00:00');

INSERT INTO optimizacion.semestres VALUES (1, 'Primer Semestre', '1er Sem');

INSERT INTO optimizacion.anioacademico VALUES (1, 1, 'Primero');

INSERT INTO optimizacion.facultades VALUES (1, 'Ciencias');

INSERT INTO optimizacion.cursos VALUES (2, '1', ' Química General', 'FCQG');

INSERT INTO optimizacion.cursosfacultades VALUES (1, 1);

INSERT INTO optimizacion.cursosalumnos VALUES (92001, 1,5);

INSERT INTO optimizacion.examenesalu VALUES (1, 92001,1001,10);

7) Optimizador de PostgreSQL

Realizar todas las consultas del práctico. Analizar cada uno de los resultados y proponer las estructuras adicionales que crea convenientes.

Ejecutar el cálculo de estadísticas, vuelva a analizar los resultados y explique el comportamiento encontrado.

a) SELECT \* FROM EXAMENES

tiempo 0.230 s

b) SELECT NOMALU, NOMCUR

FROM ALUMNOS A, CURSOSALUMNOS CA, CURSOS C

WHERE A.IDALU=CA.IDALU

Tiempo 15.500s, 200000 recorridas de tabla

Sugerencia: create index indiceAlu on optimizacion.alumnos ("nomalu","idAlu");

Tiempo: 0.072s

c) SELECT NOMALU, NOMCUR

FROM ALUMNOS A, CURSOSALUMNOS CA, CURSOS C

WHERE A.IDALU=CA.IDALU

AND CA.IDCUR=C.IDCUR

AND C.NOMCUR LIKE ‘BASE DE DATOS%’

AND C.IDANIO=2016

AND NOTA > 61

Tiempo 0.080 s

Sugerencia: create index indiceCurAlu on optimizacion.cursosalumnos ("IdCur","idAlu");

Tiempo 0.076 s

d) SELECT NOMALU, NOMCUR

FROM ALUMNOS A, CURSOSALUMNOS CA, CURSOS C

WHERE A.IDALU=CA.IDALU

AND CA.IDCUR=C.IDCUR

AND NOT EXISTS (SELECT \* FROM EXAMENESALU EA

WHERE EA.IDCUR=C.IDCUR

AND EA.IDALU=A.IDALU

AND NOTA < 61)

Tiempo 0.126 s, 2568 consultas

Sugerencia: create index indiceExam ON optimizacion.examenesalu ("idCur", "idAlu")

Tiempo 0.076s, consultas 2568

e)SELECT \*

FROM FACULTADES F

WHERE NOT EXISTS (SELECT \* FROM CURSOSFACULTADES CF

WHERE F.IDFAC=CF.IDFAC

AND NOT EXISTS (SELECT \* FROM CURSOS C

WHERE CF.IDCUR=C.IDCUR

AND IDANIO=2016))

Tiempo 0.108 s, resultados 0

f) SELECT \*

FROM ALUMNOS A

WHERE EXISTS (SELECT \* FROM CURSOSALUMNOS CA

WHERE A.IDALU=CA.IDALU

AND NOTA >60

AND NOT EXISTS (SELECT \* FROM EXAMENESALU EA

WHERE CA.IDCUR=EA.IDCUR

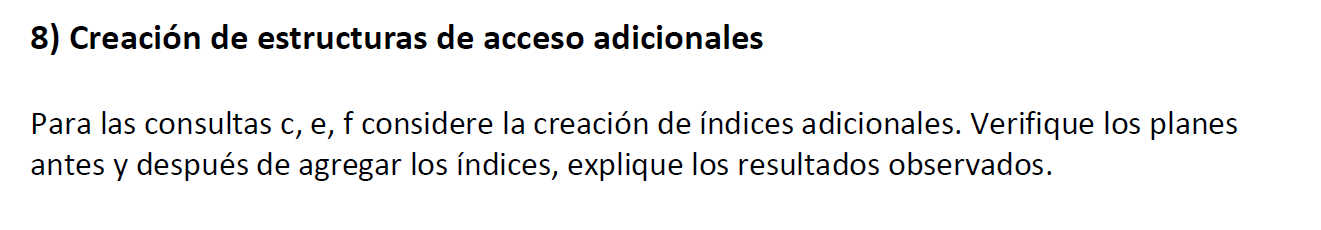
AND CA.IDALU=EA.IDALU

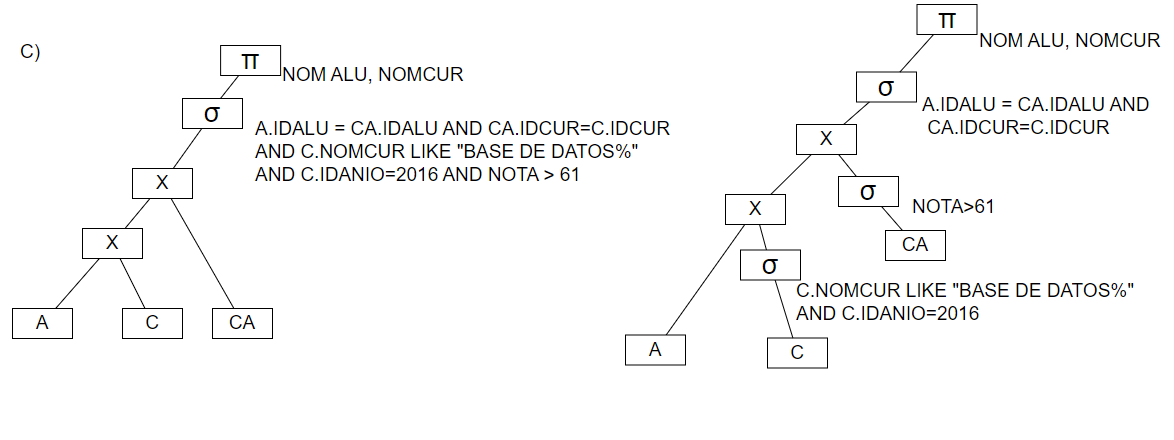
AND NOTA < 61))

Tiempo 0.083s, resultado 0

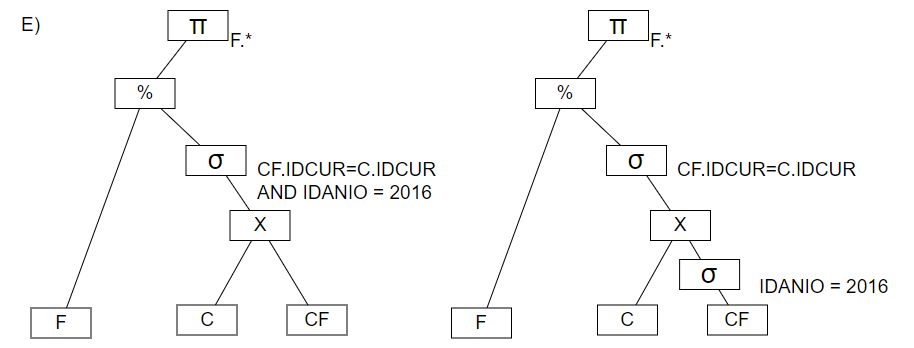
Sugerencias: create index indiceCurAluNot on optimizacion.cursosalumnos ("IdCur","idAlu", "nota");

Podria servir agregar en el modelo, una relación FACULTADES ALUMNOS a fin de ahorrar tiempo y costo de búsqueda al querer conocer información sobre los cursos o exámenes de un alumno, ya que podría buscar en esta nueva tabla mejorando la calidad de la búsqueda.

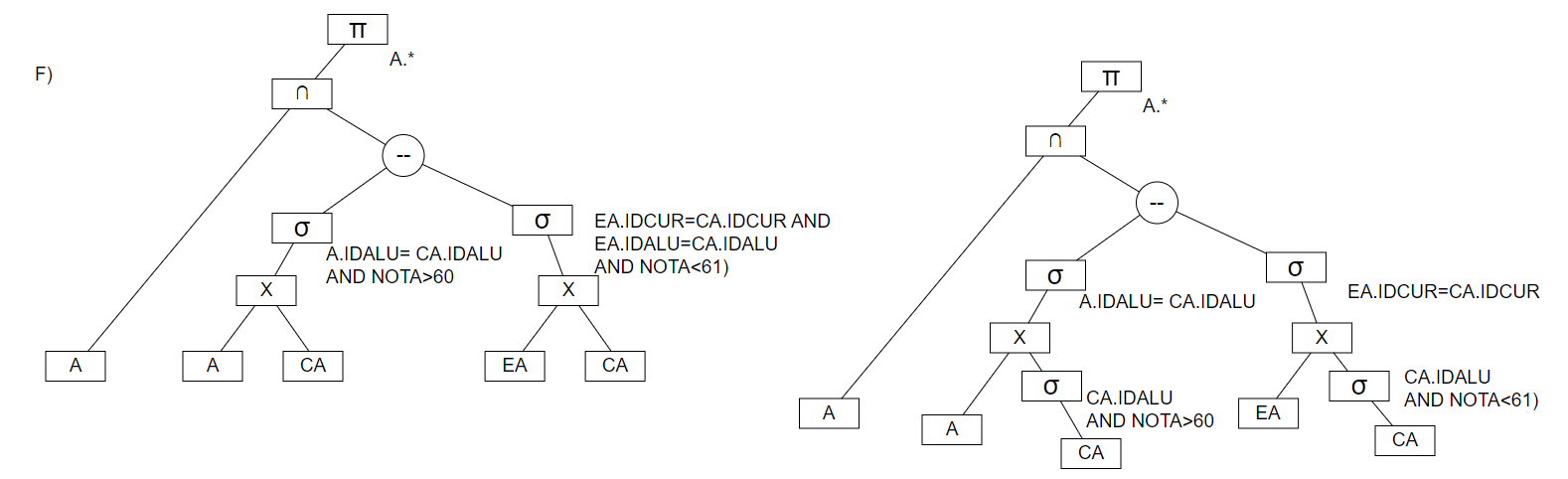




Se reduce la cantidad de consultas al ejecutar el paso en C al filtrar los nombres de cursos y el año. Y posteriormente en CA al filtrar las notas mayores de 61



Se reduce la cantidad de consultas al filtrar por año igual a 2016 en CF, reduce así el producto cartesiano con C.



Se reducen las consultas al filtrar por notas y CA.IDALU antes de unirlo con A. De igual modo se reduce la resta al filtrar por notas y CA.IDALU antes de unirlo con EA.