****

**Gestion d’un cabinet d’ assurances**

**Base de donnes 2**

**Ahmad Said (88671)**

**Ali Slim(87592)**

**Mohamad Naji (92105)**

# Objectifs :

Améliorer la productivité du cabinet d’assurances en informatisant ses données en un system de gestion de donné spécifiés aux agents. On peut faire les tâches suivantes :

* Ouverture de nouveaux contrats auprès des clients.
* Encaissement annuel des primes d’assurances souscrits par ses clients, en reversant à chacune des compagnies ce qui lui revient.
* Traitement des sinistres : créer des rapports et établir le montant de l’indemnité.

# Entité Relation :

Notre Project est base sur l’agent ou il peut faire tous opérations facilement entre le client et la compagnie associe, d’où le besoin tout d’abord de ces 3 entités.

De plus les compagnies ont de plusieurs types d’assurance qui peuvent être communs entre eux. D’où aussi une autre entité qu’est le type.

Les taches des agents sont désignées par des relations entre ces entités, la principale tâche est créée un contrat tout en satisfaisant les conditions du Project, et ces conditions sont désignés par des triggers nécessaires avant l’insertion d’un élément de cette relation.

Plus précisément, l’essentiel du temps des différents agents du cabinet est occupé par les tâches suivantes :

 Ouverture de nouveaux contrats auprès des clients. Outre la prospection de nouveaux clients, cela consiste à négocier le contenu du contrat avec le client, établir le contrat en partant de l’un des contrats types de la compagnie d’assurances choisie par le client, et le faire approuver par la compagnie. Il y a souvent des aller-retour dans cette procédure. Un client peut avoir plusieurs contrats d’assurances, mais ils doivent tous être souscrits auprès de la même compagnie. De même, un agent s’occupe de tous les contrats d’un client.

 Encaissement annuel des primes d’assurances souscrits par ses clients, en reversant à chacune des compagnies ce qui lui revient.

 Traitement des sinistres. L’agent est chargé d’évaluer le montant de l’indemnité que la compagnie versera au client pour un sinistre. Après avoir fait son évaluation, il la transmet à la compagnie qui règle directement le client. En ce qui concerne la réparation des préjudices causés par un sinistre (qu’il s’agisse de la réparation d’un bien immobilier ou d’une voiture, du rachat d’objets volés ou détériorés, ou encore de soins médicaux), le client s’en charge tout seul, il utilise comme il le veut le montant de l’indemnité versée par la compagnie.

Pour simplifier, on suppose qu’un sinistre comporte un seul préjudice et qui ne

Ne sont pas tous évalués de la même façon.

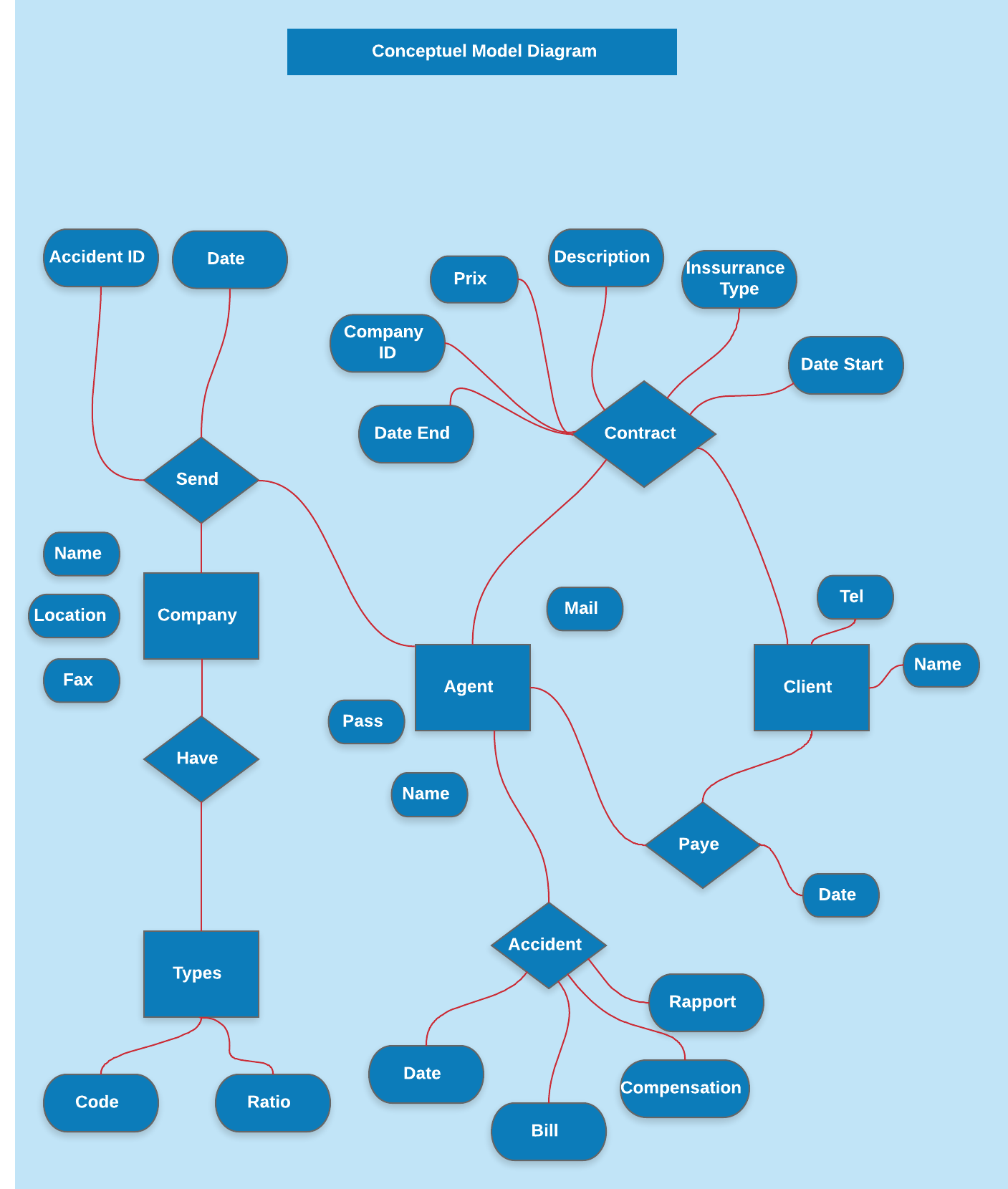
Pour établir le montant de l’indemnité, l’agent procède différemment selon les

Cas :

- soit il fait lui-même une évaluation à priori du montant des réparations, il établit alors un rapport d’expertise qu’il adresse à la compagnie, et celle-ci le transmet au client ;

- soit l’agent d’assurance calcule l’indemnité à partir de factures présentées par le client ; l’indemnité est égale à l’intégralité du montant de la facture dans certains cas (c’est notamment celui des frais médicaux) ; dans d’autres cas, on considère que le bien était déjà partiellement amorti au moment du sinistre, et le montant de l’indemnité ne couvre qu’une partie de celui de la facture.

Diagramme voire la page suivante :

Diagramme Entité-Relation : (notant que chaque entité a sa propre id qui est la clé primaire de cette dernière)

ID

1. Types Abstrait de données

create type udname from varchar(50)

create type udid from int not null

create type udmail from varchar(100)

create rule mail\_checker

as @mail like '[a-zA-Z]%@%.%'

go

--drop rule mail\_checker

exec sp\_bindrule 'mail\_checker','udmail'

go

--exec sp\_unbindrule 'udmail','mail\_checker';

go

create type temp as table(

tid udid primary key identity(1,1),

clid int,

agid int,

cpid int,

ctid int,

icid int,

itid int,

date date,

sdate date,

edate date

)

# Trigger :

## CheckValidContrat

Ce trigger est de type ‘instead of insert’ sur le tableau contrat pour être sûr que les donnes de contrat sont valides dont ils respectent les règles ci-dessous, sinon la commande d’insertion est rejetée.

Règles vérifiées avant insertion :

* À propos d’agent vérifier que le client fait un contrat avec le même agent qu’il a déjà été traite si le client est ancien.
* À propos de la compagnie vérifier que c’est la même compagnie du type de contrat.
* Vérifier aussi que cette compagnie a le type précisé dans le commande
* Vérifier que le date de début est avant la date de fin.

Code :

create trigger CheckValidContrat

on contrats

instead of insert -- data inserted form the user are in the table 'inserted'

as

begin

declare @clid int

declare @sdate date

declare @dif int,@cpid int, @agid int

declare @inscpid int, @insagid int,@valid int

declare @instype int, @insstartdate date,@insenddate date

declare @code varchar(5)

set @valid =0

-- set @clid =(select id from inserted)

select @clid=CLID,@inscpid=CPID,@insagid=AGID,@instype=ITID,@insstartdate=[Start Date],@insenddate=[End Date] from inserted

-- use later cursor to fetch all contrats and if it was with same type check same company otherwise continue

select @sdate=[End Date],@agid=AGID from Contrats where CLID=@clid order by [End Date] -- give biggest end date

if(@sdate is not null)

begin

set @dif = DATEDIFF(day,getdate(),@sdate) -- return diff days @sdate - getdate() // system database time

if(@dif>0)

begin

if(@agid != @insagid)

begin print 'error changing ag id is not allowed' end

else

if(@cpid is not null and @cpid!=@inscpid)

begin print 'error chaging company for the same type is not allowed' end

begin

set @valid =1

end

end

else

begin

set @valid =1

end

end

else

begin

set @valid =1

end

if(not exists (select ITID from CPhaveIT where CPID=@inscpid and ITID= @instype))

begin

print 'Chosen Company does not have specifeid type'

set @valid =0

end

if(DATEDIFF(day,@insstartdate,@insenddate)<0)

begin

print 'Invalid Date input'

set @valid =0

end

if(@valid =1)

begin insert into Contrats SELECT

[AGID]

,[CLID]

,[CPID]

,[ITID]

,[Start Date]

,[End Date]

,[Price]

,[Description]

FROM inserted

end

end

go

* 1. Check\_Paid\_before\_incident

Ce trigger est de type ‘instead of insert’ sur le tableau incident, il joue un rôle de vérifier que les incidents basant sur le contrat spécifié est valable c.à.d. Que la date de l’incident est incluse dans l’intervalle du contrat de plus que le client a payé tous ses créances financières jusqu’à le moment de l’incident.

Code :

create trigger Check\_Paid\_before\_incident

on incidents

instead of insert

as

begin

declare @CLID int, @CTID int,@difyear int, @insDate date, @instype int

declare @EndDate date, @StartDate date, @dif int, @valid int, @paid int

declare @bill float

set @valid = 1

select @CTID = CTID,@insDate=Date,@bill=CLBILL from inserted

select @EndDate =[End Date], @StartDate = [Start Date], @instype=ITID from Contrats where id=@CTID

--set @dif = DATEDIFF(day,getdate(),@EndDate)

if(not(datediff(day,@startdate, @insdate)>0 and datediff(day,@insdate,@EndDate)>0)) begin set @valid = 0 print 'Contrat does not include that date' end

--set @dif = DATEDIFF(MONTH,@StartDate,GETDATE())

set @dif = DATEDIFF(MONTH,@StartDate,@insDate)

set @difyear= CEILING(@dif/12.0)

set @paid = dbo.paid(@CTID)

if(not(@paid >= @difyear or ( @paid= @difyear -1 and @dif%12=1) ))

begin

print 'you have not paid all year contrats broken' -- we can add broken status in contrat

set @valid =0

end

if(@valid=1)

begin

declare @code varchar(5),@ratio float,@comp float

select @code= code,@ratio=ratio from InsuranceTypes where id=@instype

if(@code='MED')

begin set @comp = @ratio \* @bill /100 end

INSERT INTO [dbo].[Incidents]

select

[AGID]

,[CLID]

,[CTID]

,[Date]

,[CLBILL]

,@comp

,[Rapport]

from inserted

end

end

go

1. Procédure et Fonctions
   1. ClientReport

Ce Procédure prend un paramètre l’ID du client et sert à afficher toutes les informations correspondantes telles que son profile, ses contrats et le nombre de fois il a payé pour ce contrat (en utilisant la fonction paid) et la compagnie correspondant avec le type d’assurance correspondant.

Code :

create procedure ClientReport @clid udid

as

declare contrat\_cursor cursor for

SELECT [ID]

,[AGID]

,[CLID]

,[CPID]

,[ITID]

,[Start Date]

,[End Date]

,[Price]

FROM [dbo].[Contrats] where [CLID]=@clid

declare @ctid int

declare @agid int,@agname udname,@aglast udname

declare @cpid int,@cpname udname

declare @itid int, @instype varchar(5), @ratio float

declare @sdate date, @edate date

declare @price float

declare @paid int

declare @print ClientRP

open contrat\_cursor

fetch next from contrat\_cursor into @ctid,@agid,@clid,@cpid,@itid,@sdate,@edate,@price

while @@FETCH\_STATUS =0

begin

select @agname=Name, @aglast=Last from Agent

select @cpname=Name from Company

select @instype=Code, @ratio=Ratio from InsuranceTypes

select @paid= dbo.paid(@ctid);

insert into @print values(@ctid,@instype,@ratio,@price,@sdate,@edate,@paid,@cpid,@cpname,@agid,@agname,@aglast)

fetch next from contrat\_cursor into @ctid,@agid,@clid,@cpid,@itid,@sdate,@edate,@price

end

close contrat\_cursor

deallocate contrat\_cursor

select \* from Client where id=@clid

select \* from @print

go

* 1. ClientDept

Pour faciliter le travail pour un agent c’est procédure qui représente tous les clients qui doivent payer jusqu’à notre moment en précisant l’ID du client, le contrat, et en calculant l’année manquante.

create procedure ClientDebt

--returns table

as

declare @clid int

declare @print ClientDept

declare id\_curse cursor for

select id from dbo.Client

open id\_curse

fetch next from id\_curse into @clid

while @@FETCH\_STATUS =0

begin

declare contrat\_cursor cursor for

select id,agid,[Start Date],[End Date] from Contrats where CLID=@CLID

declare @ctid int,@agid int

declare @sdate date,@edate date

declare @paid int

open contrat\_cursor

fetch contrat\_cursor into @ctid,@agid,@sdate,@edate

while @@FETCH\_STATUS = 0

begin

-- later do change end date in contrats to period of year

set @paid=dbo.paid(@ctid)

set @sdate=DATEADD(year,@paid,@sdate)

declare @i int

while(DATEDIFF(year,@sdate,getdate())>=0)

begin

insert into @print values(@agid,@clid,@ctid,@sdate)

set @sdate=DATEADD(year,1,@sdate)

end

fetch contrat\_cursor into @ctid,@agid,@sdate,@edate

end

close contrat\_cursor

deallocate contrat\_cursor

fetch next from id\_curse into @clid

end

close id\_curse

deallocate id\_curse

select \* from @print go

-----

* 1. Paid Function

Retourne le nombre de fois des payements par le client pour un contrat pris l’id du dernier comme paramètre.

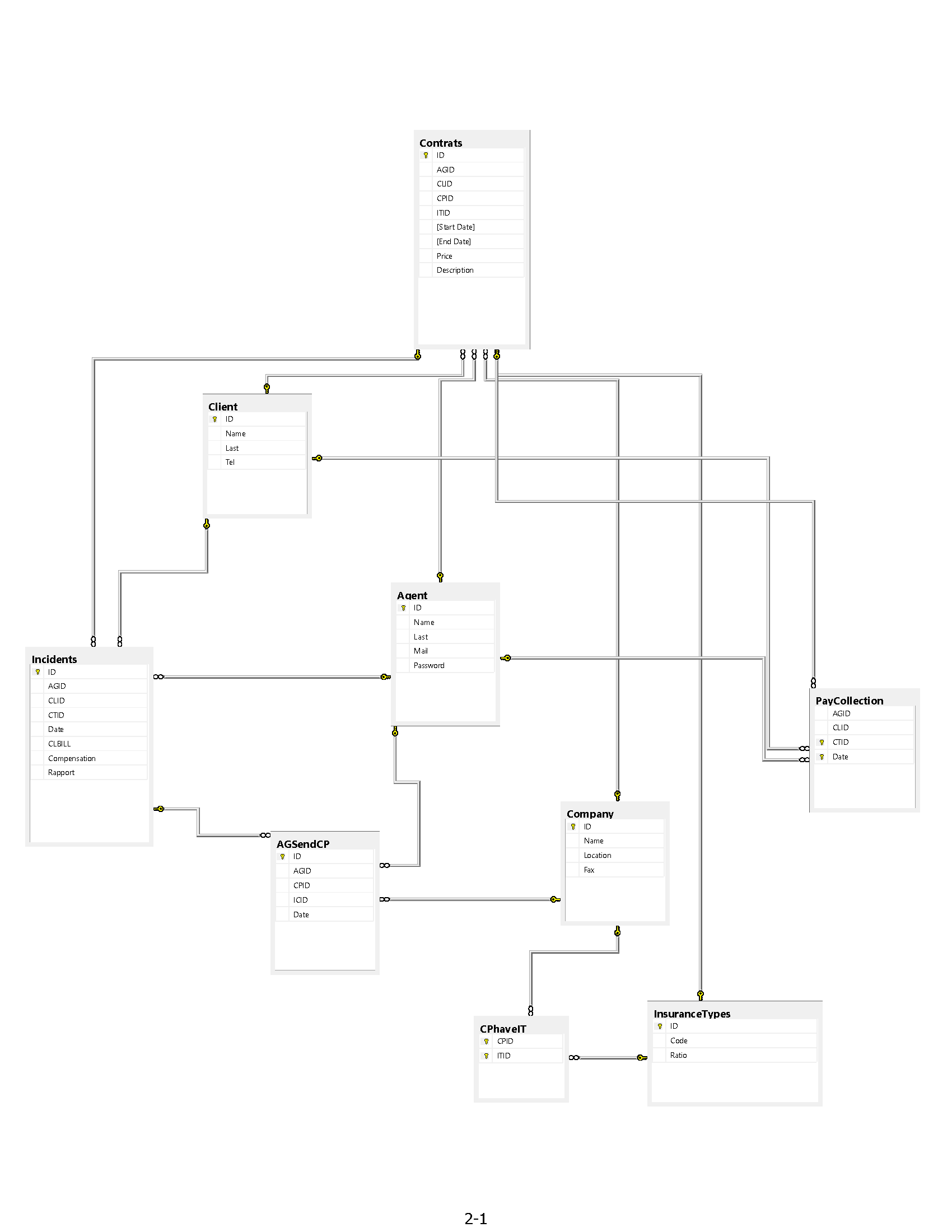
create function paid (@CTID int)

returns int

begin

return (select count(\*) from PayCollection where CTID=@ctid)

end

Schéma Relationnel