Héritage

IMAGE HERITAGE

1. Rappel

IMAGE RAPPEL

Signifie que `B` herite de `A` et suppose que l'on peut dire `B` est un `A`

NE JAMAIS UTILISER L'HÉRITAGE DE POSTGRESql

1.1. Tradution:

```
Oeuvre(*id_oeuvre, titre)
Film(*id_oeuvre#, duree)
Lvre(*id_oeuvre#, nb_page)
Auteur(*(id_oeuvre#, id_artiste#))  /*id_oeuvre réference a oeuvre*/
JoueDans(*(id_oeuvre#, id_artiste#)) /*id_oeuvre réference a film*/
```

Création de table dans PostgreSql

1. Syntaxe general

```
CREAT_TABLE nom_Table (
    nom_attr1   type_attr1 [CONSTRAINT attr1]
    ...
    nom_attrn   type_attrn [CONSTRAINT attrn]

    CONSTRAINT ligne_1,
    ..
    CONSTRAINT ligne_n
    );
```

1.1. Exemple

```
CREAT_TABLE artiste (

    id_artiste int PRIMARY KEY,
    nom_artiste text NOT NULL,
    prenom_artiste text,
    annee_de_naissance date
    );
```

FILM	
1	2h30
...	...

OEUVRE	
1	Titanic
2	La chute
3	Guernica
...	...

LIVRE	
2	100
...	...

2. Les type principaux

- `int` : entier
- `bool` : boolean
- `real` : réel
- `numeric (precision, echelle)` : nombre décimaux
    - precision: nombre de chiffre au total
    - echelle : après la virgule
- `texte` : chaine de charactère
- `varchar(longueur)` :
    - chaine de charactère de longueur 'longueur'
- `date` : date
- `time` : heure
- `timestamp` : date et heure
- `serial` : entier à incrementation automatique (permet de donner des identifiants tous different)

2.1. Contrainte

Elle sont vérifiées lors de : * ``INSERT `(ajout de ligne) *` UPDATE `(modification) *` DELETE```(suppression)

Clef primaire

Syntaxe

A la fin de la declaration de l'attribut : `PRIMARY KEY` (si clef primaire = 1 seul attribut) en contrainte de ligne

```
[CONSTRAINT nom_contrainte] PRIMARY KEY (attr1,attr2,...,attrn)
```

Exemple:

Chambre d'hotel

```
PRIMARY KEY (id_hotel, num_chambre)
```

```
base
  |
  |---- Schema (un schema par étudiant
```

On ne peut avoir 2 fois le même nom de contrainte dans le même schema. Il est donc préférable de préfixer le nom de la contrainte par le nom de la table

Si on ne donne pas de nom `Postgres` le fait:

```
\d nom_table
```

- Une clef primaire ne peut jamais être `NULL`. Idem pour un des attribut qui la compose
- `Postgres` vérifie lors de l' `ÌNSERT``` et ÙPDATE``` avec la clef n'est pas déja dans la base

Clefs étrangères

Apres la déclaration d'attribut

```
REFERENCES nom_table [(attr)]
```

Si t [(attr)] n'est pas mise on se réfère à la clef primaire de la table

Exemple

Chambre d'hotel

```
id_hotel int REFERENCES hotel
```

Dans film (avec remake)

```
CREAT TABLE Film(
  id_film int PRIMARY KEY,
  id_origine int REFERENCE film,
  )
```

Si en contrainte de ligne

```
[CONSTRAINT nom_table]
FOREIGN KEY (attr1,...,attrn)
REFERENCES nom_table
[(attr,...,attrn)]
```

```
A(*(a,b),c)
B(*d, (b,c)#)
```

```
FOREIGN KEY (b,c) reference A : relira B(b,c) vers A(a,b)
FOREIGN KEY (b,c) reference A(b,c) B(a,b) C A(b,c)
````
```

CREATE TABLE reservation( date date, nom_chambre int, id_hotel int, id_client int REFERENCES client PRIMARY KEY (date, num_cambre, id_hotel), FOREIGN KEY (num_chambre, id_hotel) REFERENCES Chambre ) Chambre( id_hotel int REFERECES hotel )

```
```

ON DELETE 'comportement' ON UPDATE 'comportement'

```
```

A(*(a,b),c) B(*d, (a,b)#)

```
```

CREAT TABLE B(...

FOREIGN KEY (a,b) REFERENCES A ON DELETE comp1 ON UPDATE comp2)
```

- Si `comp1 = CASCADE` Si on supprime une ligne dans `A``` on va supprimer les lignes de` B `correpondantes. Si` UPDATE``, modification dans `B`
- Si `comp1 = NO ACTION` Comportement par defaut refuse le `DELETE` ou le `UPDATE` sur `A` si les ligne `B` référence la ligne `A` en question ``UPDATE `: le comprortement ne joue que si changement dans les attributs de` A `référencés par` B```
- Si `comp1 = SET NULL` Met a `NULL` les attributs référencés