<u>Dépendance fonctionnelle (DF) :</u> pour toute valeur de A une seule valeur de B lui est associée.
Ça se note A -> B.
On dit que A détermine B.
<u>ex:</u> Num_etu -> Prenom
mais Nom -/- Prenom

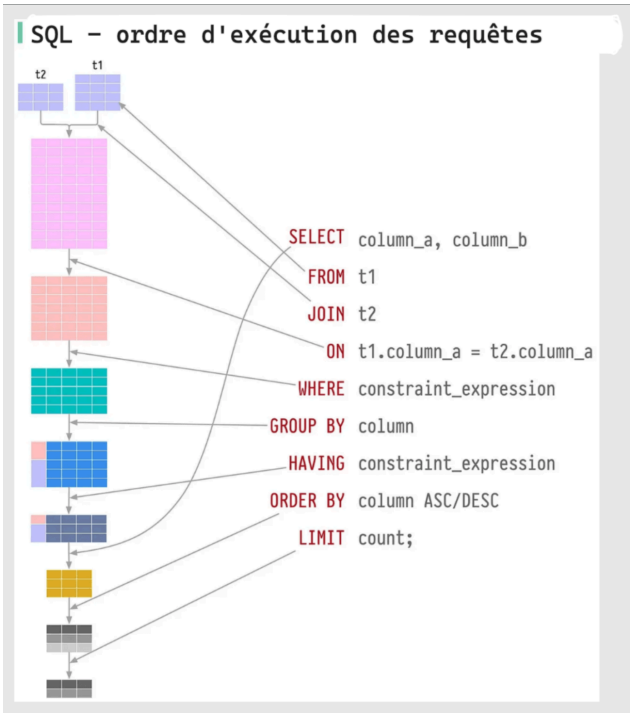
Une dépendante A->B n'est **pas directe** s'il existe un C tq A->C et C->B.

Une dépendance fonctionnelle A->B est **élémentaire** si pour toute partie A'cA, A'-/-B.

La **fermeture** permet de déterminer la clé d'une relation.

Formes normales (FN) :
1FN = attributs atomiques(il n'y a qu'une valeur a un moment donné) + clé
2FN = les attributs doivent dépendre de toute la clé
3FN = tous les attributs doivent dépendre directement de la clé

SELECT (DISTINCT) (AS)
FROM
JOIN
ON
WHERE
GROUP BY (MIN, MAX, COUNT, SUM, AVG)
HAVING (UNION, INTERSECT, EXCEPT)
ORDER BY



PLSQL :

Structure d'un bloc :

[**<Entête de bloc>**] (valable pour les fonctions, procédures, packages)
[**DECLARE**
constantes,
variables,
cursors]
BEGIN
instructions – partie exécution
EXCEPTION
gestion des erreurs
END;

DBMS_OUTPUT.PUT_LINE(variable) : affiche le contenu de la variable

Type de données :

- Types scalaires
  - **CHAR(taille)** : chaine de caractère de longeur fixe, 2000 max
  - **VARCHAR2(taille)** : chaine de caractère de longeur variable (4000 max)
  - NCHAR et NVARCHAR2 : pour les caractères unicode
  - **NUMBER** : numérique positif et négatif. A pour sous type INT, SMALLINT, REAL, DECIMAL
  - **DATE**
  - **BOOLEAN** : TRUE, FALSE
- Type implicite : fait référence à une entité déjà existante.
  - **%TYPE** : permet de faire référence à un type existant
    - ex: V_variable Etudiant.Name%type
  - **%ROWTYPE** : permet de faire référence à la structure d'une table existante
- Types définis par l'utilisateur
    - ex: **DECLARE SUBTYPE** Type_Date **IS** DATE;
- Types de données composés
  - **TABLE** : table d'éléments de même type, chq élément ayant un indice
    - DECLARE **TYPE** Nom_Type
    - IS TABLE
    - **OF** type_donnee [NOT NULL]
    - **INDEX BY** [BINARY_INTEGER | PLS_INTEGER | VARCHAR2(size limit)]
    - ex: DECLARE TYPE Type_Table IS TABLE OF VARCHAR2(50) INDEX BY BINARY_INTEGER;
  - **RECORD** : un peu comme un objet en Java,
    - ex: DECLARE TYPE Type_Record IS RECORD(champ1 type1, champ2 type2, …, champN typeN);

SELECT **sysdate** FROM **DUAL** : pour obtenir la date

```
DECLARE V_nbre NUMBER:=0;
BEGIN
FOR V_nbre in 1..3
LOOP
DBMS_OUTPUT.PUT_LINE(V_nbre);
END LOOP;
DBMS_OUTPUT.PUT_LINE('Fin de la boucle');
END
```

sort :

1
2
Fin de la boucle