

Aspect historique

En 1970, [Edgar Codd](#) invente le *modèle relationnel* pour les bases de données.

Aspect historique

En 1970, [Edgar Codd](#) invente le *modèle relationnel* pour les bases de données.
En 1973, première version du langage **SQL** (**S**tructured **Q**uery **L**anguage).
C'est un langage de requêtes permettant d'interagir avec une base de données.

C2 Bases de données et SQL

Aspect historique

En 1970, [Edgar Codd](#) invente le *modèle relationnel* pour les bases de données.

En 1973, première version du langage **SQL** (**S**tructured **Q**uery **L**anguage).

C'est un langage de requêtes permettant d'interagir avec une base de données.

En 1979, apparition du premier **S**ystème de **G**estion de **B**ase de **D**onnées (sgbd) : Oracle qui fera la fortune de ses trois fondateurs (L. Ellison, E. Oats et B. Miner).

C2 Bases de données et SQL

Aspect historique

En 1970, [Edgar Codd](#) invente le *modèle relationnel* pour les bases de données.

En 1973, première version du langage **SQL** (**S**tructured **Q**uery **L**anguage).
C'est un langage de requêtes permettant d'interagir avec une base de données.

En 1979, apparition du premier **S**ystème de **G**estion de **B**ase de **D**onnées (sgbd) : Oracle qui fera la fortune de ses trois fondateurs (L. Ellison, E. Oats et B. Miner).

En 1995 (et 1996), première version d'importants sgbd [libres](#) MySQL (et PostgreSQL).

C2 Bases de données et SQL

Aspect historique

En 1970, [Edgar Codd](#) invente le *modèle relationnel* pour les bases de données.
En 1973, première version du langage **SQL** (**S**tructured **Q**uery **L**anguage).
C'est un langage de requêtes permettant d'interagir avec une base de données.

En 1979, apparition du premier **S**ystème de **G**estion de **B**ase de **D**onnées (sgbd) : Oracle qui fera la fortune de ses trois fondateurs (L. Ellison, E. Oats et B. Miner).

En 1995 (et 1996), première version d'importants sgbd [libres](#) MySQL (et PostgreSQL).

En 2020, Le volume mondial de données stockées est estimé à 47 milliards de téra-octets (47×10^{21} octets) et a été multiplié par 20 en 10 ans.

Apport des bases de données

Les bases de données corrigent de nombreux défauts des outils traditionnels de gestion des données. En effet :

- Les accès simultanés par plusieurs programmes aux mêmes données pouvaient générer des conflits.

Apport des bases de données

Les bases de données corrigent de nombreux défauts des outils traditionnels de gestion des données. En effet :

- Les accès simultanés par plusieurs programmes aux mêmes données pouvaient générer des conflits.

- Pour lire (ou modifier) les données il fallait savoir comment elles étaient représentées.

Apport des bases de données

Les bases de données corrigent de nombreux défauts des outils traditionnels de gestion des données. En effet :

- Les accès simultanés par plusieurs programmes aux mêmes données pouvaient générer des conflits.

- Pour lire (ou modifier) les données il fallait savoir comment elles étaient représentées.

- Les utilisateurs devaient s'assurer de l'intégrité des données avant de les stocker. C'est à dire que c'est l'utilisateur qui était chargé du contrôle de la validité de ses données.

C2 Bases de données et SQL

Principes des bases de données

Plusieurs aspects des bases de données viennent corriger les limitations des outils traditionnels :

Principe d'**unicité** : un enregistrement doit être unique (une donnée qui apparaît plusieurs fois est dite *redondante*).

C2 Bases de données et SQL

Principes des bases de données

Plusieurs aspects des bases de données viennent corriger les limitations des outils traditionnels :

Principe d'**unicité** : un enregistrement doit être unique (une donnée qui apparaît plusieurs fois est dite *redondante*).

Ici intervient la notion de **clé primaire**, c'est à dire dans une table une identification unique de l'enregistrement.

C2 Bases de données et SQL

Principes des bases de données

Plusieurs aspects des bases de données viennent corriger les limitations des outils traditionnels :

Principe d'**unicité** : un enregistrement doit être unique (une donnée qui apparaît plusieurs fois est dite *redondante*).

Ici intervient la notion de **clé primaire**, c'est à dire dans une table une identification unique de l'enregistrement.

Principe d'**intégrité** : le contrôle de la validité des données est effectué par le sgbd.

C2 Bases de données et SQL

Principes des bases de données

Plusieurs aspects des bases de données viennent corriger les limitations des outils traditionnels :

Principe d'**unicité** : un enregistrement doit être unique (une donnée qui apparaît plusieurs fois est dite *redondante*).

Ici intervient la notion de **clé primaire**, c'est à dire dans une table une identification unique de l'enregistrement.

Principe d'**intégrité** : le contrôle de la validité des données est effectué par le sgbd.

Ici intervient la notion de **domaine**, c'est à dire qu'on peut préciser que les valeurs d'un champ doivent être d'un certain types (par exemple entier, flottant, chaîne de caractères, ...) et appartenir à un certain ensemble de valeurs : le domaine.

C2 Bases de données et SQL

Principes des bases de données

Plusieurs aspects des bases de données viennent corriger les limitations des outils traditionnels :

Principe d'**unicité** : un enregistrement doit être unique (une donnée qui apparaît plusieurs fois est dite *redondante*).

Ici intervient la notion de **clé primaire**, c'est à dire dans une table une identification unique de l'enregistrement.

Principe d'**intégrité** : le contrôle de la validité des données est effectué par le sgbd.

Ici intervient la notion de **domaine**, c'est à dire qu'on peut préciser que les valeurs d'un champ doivent être d'un certain types (par exemple entier, flottant, chaîne de caractères, ...) et appartenir à un certain ensemble de valeurs : le domaine.

Principe d'**indépendance logique** : les utilisateurs accèdent aux données sans se soucier de la façon dont elles sont représentées ou codées dans la base.

C2 Bases de données et SQL

Exemple

Prenons l'exemple suivant :

Nom	Prénom	Naissance
Pascal	Blaise	1623
Lovelace	Ada	1815
Boole	George	1815

Il est certes peu probable (mais pas impossible) que deux personnes portant les mêmes noms et prénoms naissent la même année, afin de respecter le **principe d'unicité**, nous devons adjoindre à chaque enregistrement un champ (par exemple `id`) unique qui servira de clé primaire.

C2 Bases de données et SQL

Exemple

Prenons l'exemple suivant :

Nom	Prénom	Naissance
Pascal	Blaise	1623
Lovelace	Ada	1815
Boole	George	1815

Il est certes peu probable (mais pas impossible) que deux personnes portant les mêmes noms et prénoms naissent la même année, afin de respecter le **principe d'unicité**, nous devons adjoindre à chaque enregistrement un champ (par exemple `id`) unique qui servira de clé primaire.

Les champs `Nom` et `Prénom` sont au format texte, le champ `Naissance` est un entier.

C2 Bases de données et SQL

Exemple

Prenons l'exemple suivant :

Nom	Prénom	Naissance
Pascal	Blaise	1623
Lovelace	Ada	1815
Boole	George	1815

Il est certes peu probable (mais pas impossible) que deux personnes portant les mêmes noms et prénoms naissent la même année, afin de respecter le **principe d'unicité**, nous devons adjoindre à chaque enregistrement un champ (par exemple id) unique qui servira de clé primaire.

Les champs Nom et Prénom sont au format texte, le champ Naissance est un entier.

On peut par exemple préciser les contraintes d'intégrité suivantes : Nom doit être non vide, Naissance doit être supérieur à 0.

C2 Bases de données et SQL

Exemple

Prenons l'exemple suivant :

Nom	Prénom	Naissance
Pascal	Blaise	1623
Lovelace	Ada	1815
Boole	George	1815

Il est certes peu probable (mais pas impossible) que deux personnes portant les mêmes noms et prénoms naissent la même année, afin de respecter le **principe d'unicité**, nous devons adjoindre à chaque enregistrement un champ (par exemple id) unique qui servira de clé primaire.

Les champs Nom et Prénom sont au format texte, le champ Naissance est un entier.

On peut par exemple préciser les contraintes d'intégrité suivantes : Nom doit être non vide, Naissance doit être supérieur à 0.

Nous faisons référence à cette base sous le nom **exemple** dans la suite

C2 Bases de données et SQL

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

C2 Bases de données et SQL

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table `personne` peut s'écrire :

C2 Bases de données et SQL

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table `personne` peut s'écrire :

Sous forme de liste :

C2 Bases de données et SQL

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table `personne` peut s'écrire :

Sous forme de liste :

personnes (id : int, Nom : text, Prenom : text, Naissance : int)

C2 Bases de données et SQL

Schéma relationnel

Le schéma relationnel d'une base de données présente les tables de cette base sous la forme de liste ou de tableau. Dans les deux cas, on précise la clé primaire de la table en soulignant l'attribut. On indique aussi parfois le type des attributs.

Exemple

Le schéma relationnel de la table personne peut s'écrire :

Sous forme de liste :

personnes (id : int, Nom : text, Prenom : text, Naissance : int)

Sous forme de tableau :

personnes	
<u>id</u>	: int
Nom	: text
Prenom	: text
Naissance	: int

C2 Bases de données et SQL

Premiers pas en SQL

Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :

Exemples

C2 Bases de données et SQL

Premiers pas en SQL

Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :
`select * from table`

Exemples

C2 Bases de données et SQL

Premiers pas en SQL

Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :
`select * from table`

Pour récupérer simplement les champs `champ1`, `champ2`, ... on utilise :

Exemples

C2 Bases de données et SQL

Premiers pas en SQL

Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :
`select * from table`

Pour récupérer simplement les champs `champ1`, `champ2`,... on utilise :
`select champ1, champ2,... from table`

Exemples

C2 Bases de données et SQL

Premiers pas en SQL

Pour récupérer la totalité des champs d'une table `table` on utilise la syntaxe :
`select * from table`

Pour récupérer simplement les champs `champ1`, `champ2`,... on utilise :
`select champ1, champ2,... from table`

Exemples

```
select Nom, Naissance from exemple
```

Pascal	1623
Lovelace	1815
Boole	1815

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certaines conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Exemples

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certaines conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **between** (entre)

Exemples

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certaines conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **between** (entre)

Logique : **and**, **or** et **not**

Exemples

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **between** (entre)

Logique : **and**, **or** et **not**

Modèle de chaînes de caractères : **like** où **%** désigne n'importe quel suite de caractères et **_** un unique caractère.

Exemples

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **between** (entre)

Logique : **and**, **or** et **not**

Modèle de chaînes de caractères : **like** où **%** désigne n'importe quel suite de caractères et **_** un unique caractère.

Exemples

Pour chercher dans la table les personnes nées après 1789 :

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certaines conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **between** (entre)

Logique : **and**, **or** et **not**

Modèle de chaînes de caractères : **like** où **%** désigne n'importe quel suite de caractères et **_** un unique caractère.

Exemples

Pour chercher dans la table les personnes nées après 1789 :

```
select * from exemple where naissance > 1789
```

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certaines conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **between** (entre)

Logique : **and**, **or** et **not**

Modèle de chaînes de caractères : **like** où **%** désigne n'importe quel suite de caractères et **_** un unique caractère.

Exemples

Pour chercher dans la table les personnes nées après 1789 :

```
select * from exemple where naissance > 1789
```

Pour chercher dans la table les personnes dont la deuxième lettre du nom est un e :

C2 Bases de données et SQL

Clause where

Une instruction **select** peut être suivie d'une clause **where** qui permet de rechercher les enregistrements correspondants à certaines conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

Comparaison : **=**, **<**, **>**, **<=**, **>=**, **<>** (différent) et **between** (entre)

Logique : **and**, **or** et **not**

Modèle de chaînes de caractères : **like** où **%** désigne n'importe quel suite de caractères et **_** un unique caractère.

Exemples

Pour chercher dans la table les personnes nées après 1789 :

```
select * from exemple where naissance > 1789
```

Pour chercher dans la table les personnes dont la deuxième lettre du nom est un e :

```
select * from exemple where nom like "_e%"
```

C2 Bases de données et SQL

Clause order by

Une instruction `select` peut être suivie d'une clause `order by` qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle même suivie de :

Exemples

C2 Bases de données et SQL

Clause order by

Une instruction `select` peut être suivie d'une clause `order by` qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle même suivie de :

`asc` pour indiquer un classement par ordre croissant

Exemples

C2 Bases de données et SQL

Clause order by

Une instruction `select` peut être suivie d'une clause `order by` qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle même suivie de :

`asc` pour indiquer un classement par ordre croissant

`desc` pour indiquer un classement par ordre décroissant

Exemples

C2 Bases de données et SQL

Clause order by

Une instruction `select` peut être suivie d'une clause `order by` qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle même suivie de :

- `asc` pour indiquer un classement par ordre croissant

- `desc` pour indiquer un classement par ordre décroissant

La valeur par défaut est `asc`

Exemples

C2 Bases de données et SQL

Clause order by

Une instruction `select` peut être suivie d'une clause `order by` qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle même suivie de :

- `asc` pour indiquer un classement par ordre croissant

- `desc` pour indiquer un classement par ordre décroissant

La valeur par défaut est `asc`

Exemples

Pour classer par ordre alphabétique nom puis prénom notre table exemple :

C2 Bases de données et SQL

Clause order by

Une instruction `select` peut être suivie d'une clause `order by` qui permet de classer les enregistrements selon un ou plusieurs champs. Cette clause est elle même suivie de :

- `asc` pour indiquer un classement par ordre croissant

- `desc` pour indiquer un classement par ordre décroissant

La valeur par défaut est `asc`

Exemples

Pour classer par ordre alphabétique nom puis prénom notre table exemple :

```
select * from exemple order by Nom, Prenom asc
```

C2 Bases de données et SQL

Clause distinct et limit

Exemples

C2 Bases de données et SQL

Clause distinct et limit

Une instruction `select` peut être *directement* suivie d'une clause `distinct` `champ` qui indique que `champ` ne doit apparaître qu'une fois dans les résultats

Exemples

C2 Bases de données et SQL

Clause distinct et limit

Une instruction `select` peut être *directement* suivie d'une clause `distinct` `champ` qui indique que `champ` ne doit apparaître qu'une fois dans les résultats

Une instruction `select` peut être suivie d'une clause `limit` qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec `order by`.

Exemples

C2 Bases de données et SQL

Clause distinct et limit

Une instruction `select` peut être *directement* suivie d'une clause `distinct` `champ` qui indique que `champ` ne doit apparaître qu'une fois dans les résultats

Une instruction `select` peut être suivie d'une clause `limit` qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec `order by`.

Exemples

C2 Bases de données et SQL

Clause distinct et limit

Une instruction `select` peut être *directement* suivie d'une clause `distinct` `champ` qui indique que `champ` ne doit apparaître qu'une fois dans les résultats

Une instruction `select` peut être suivie d'une clause `limit` qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec `order by`.

Exemples

Pour afficher les années de naissance sans répétitions :

C2 Bases de données et SQL

Clause distinct et limit

Une instruction `select` peut être *directement* suivie d'une clause `distinct` `champ` qui indique que `champ` ne doit apparaître qu'une fois dans les résultats

Une instruction `select` peut être suivie d'une clause `limit` qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec `order by`.

Exemples

Pour afficher les années de naissance sans répétitions :

```
select distinct naissance from exemple
```

C2 Bases de données et SQL

Clause distinct et limit

Une instruction `select` peut être *directement* suivie d'une clause `distinct` champ qui indique que champ ne doit apparaître qu'une fois dans les résultats

Une instruction `select` peut être suivie d'une clause `limit` qui indique le nombre maximal d'enregistrement à renvoyer. Cette clause est particulièrement utile en relation avec `order by`.

Exemples

Pour afficher les années de naissance sans répétitions :

```
select distinct naissance from exemple
```

Pour afficher les trois plus jeunes personnes de la table :

```
select * from exemple order by naissance desc limit 3
```


C2 Bases de données et SQL

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

Exemples

C2 Bases de données et SQL

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

min pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

Exemples

C2 Bases de données et SQL

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

min pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

max pour obtenir le max

Exemples

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- min pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

- max pour obtenir le max

- sum pour obtenir la somme

Exemples

C2 Bases de données et SQL

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- min pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

- max pour obtenir le max

- sum pour obtenir la somme

- avg pour obtenir le minimum

Exemples

C2 Bases de données et SQL

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- min pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

- max pour obtenir le max

- sum pour obtenir la somme

- avg pour obtenir le minimum

- count pour compter le nombre d'enregistrement

Exemples

C2 Bases de données et SQL

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- min pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

- max pour obtenir le max

- sum pour obtenir la somme

- avg pour obtenir le minimum

- count pour compter le nombre d'enregistrement

Exemples

Pour avoir la personne la plus âgée présente dans la table :

C2 Bases de données et SQL

Agrégation

Le langage sql offre des opérateurs appelés **fonction d'agrégation** permettant de calculer une valeur à partir d'un ensemble d'enregistrement :

- min pour obtenir le minimum (d'un champ sur un ensemble d'enregistrement)

- max pour obtenir le max

- sum pour obtenir la somme

- avg pour obtenir le minimum

- count pour compter le nombre d'enregistrement

Exemples

Pour avoir la personne la plus âgée présente dans la table :

```
select min(Naissance), Nom, Prénom from exemple
```