C2 Bases de données et SQL

Aspect historique

- En 1970, Edgar Codd invente le *modèle relationnel* pour les bases 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.

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.

C2 Bases de données et SQL

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.

C2 Bases de données et SQL

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 invervient 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 invervient la notion de clé primaire, c'est à dire dans une table une identification unique de l'enregistrement.
- Principe d'intégrite : 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 invervient la notion de clé primaire, c'est à dire dans une table une identification unique de l'enregistrement.
- Principe d'intégrite : 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, chaine 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 inverviient la notion de clé primaire, c'est à dire dans une table une identification unique de l'enregistrement.
- Principe d'intégrite : 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, chaine 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.

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.

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.

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.

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

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

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 :

Exemples

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)

Exemples

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`

Exemples

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 chaines de caractères : LIKE où `%` désigne n'importe quel suite de caractères et `_` un unique caractère.

Exemples

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 chaines 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 :

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 chaines 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 certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

- Comparaison : `=`, `<`, `>`, `<=`, `>=`, `<>` (différent) et BETWEEN (entre)
- Logique : `and`, `or` et `not`
- Modèle de chaines 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 certains conditions. Ces conditions s'expriment à l'aide des opérateurs suivant :

- Comparaison : `=`, `<`, `>`, `<=`, `>=`, `<>` (différent) et BETWEEN (entre)
- Logique : `and`, `or` et `not`
- Modèle de chaines 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%"`

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

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

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

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

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 :

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

Clause DISTINCT et LIMIT

Exemples

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

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

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

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 :

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`

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

Agrégation

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

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)

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

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

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

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

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`