Теоретические основы языков манипулирования данными

DML

Создали: Рахманкулов Гуванч, Сангаре Мори

Группа: ИСТ-15

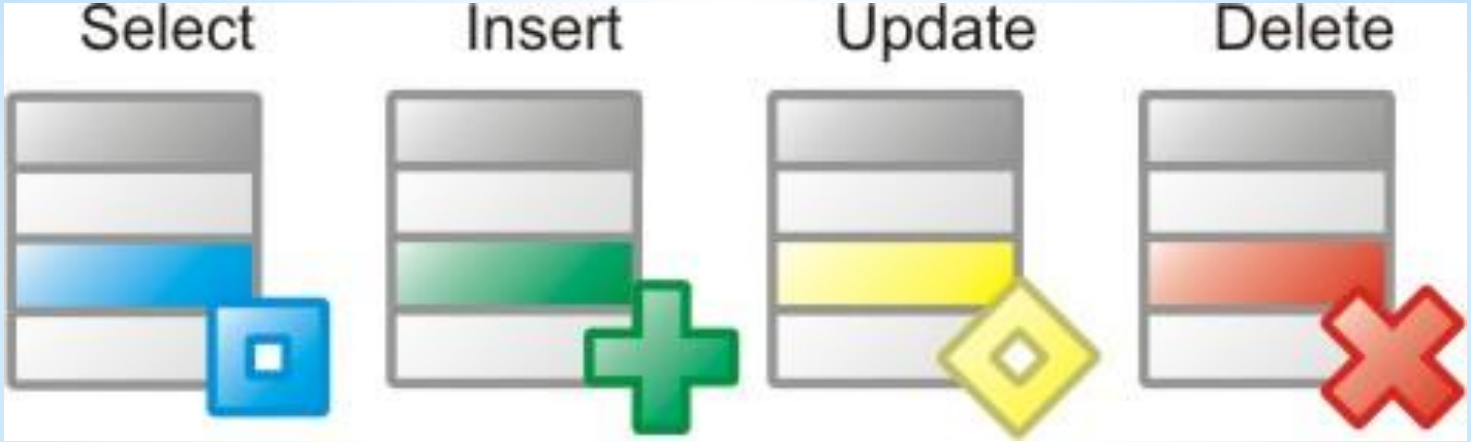
2-ой курс

DML-это...

* **Data Manipulation Language** (**DML**) (язык управления (манипулирования) данными) — это семейство компьютерных языков, используемых в компьютерных программах или пользователями баз данных для получения, вставки, удаления или изменения данных в базах данных.

Языки DML изначально использовались только компьютерными программами, но с появлением SQL стали также использоваться и людьми.

Функции языков DML определяются первым словом в предложении (часто называемом **запросом**), которое почти всегда является глаголом. В случае с SQL эти глаголы — «select» («выбрать»), «insert» («вставить»), «update» («обновить»), и «delete» («удалить»). Это превращает природу языка в ряд обязательных утверждений (команд) к базе данных.



DML (Data Manipulation Language).

- DML - Data Manipulation Language. Язык манипулирования данными. Используется для работы с информацией, хранимой в базе данных.
- Основными командами этой группы являются:
- Select - вычитка информации.
- Insert - добавление информации.
- Update - обновление информации.
- Delete - удаление информации.

Языки DML разделяются в основном на два типа:

Procedural DMLs — описывают действия над данными.

Declarative DMLs — описывают сами данные.

возможности SQL

* SQL является языком определения данных

SQL является языка определения данных (DDL), то есть сказать, что создает таблицы в реляционной базе данных, а также чем изменить или удалить.

* SQL является язык манипулирования данными

SQL является язык манипулирования данными (DML), это означает, это позволяет выбирать, вставлять, изменять или удалять данные в виде таблицы в реляционной базе данных.

* SQL является защита доступа языка

Это возможно с SQL определить разрешения на уровне пользователей базы данных. Существует говорить о DCL (Data Control Язык).

определение данными

Создание таблицы

Чтобы создать таблицу, используйте пару ключевых слов CREATE TABLE.

Синтаксис:

```
CREATE TABLE NomTable (
NomColonne1 TypeDonnée1,
NomColonne2 TypeDonnée2,
...
);
```

Вставка строк для создания

Можно создать таблицу, непосредственно вставив линии во время создания. Восстанавливаются должны быть вставлены с AS SELECT.

```
CREATE TABLE NomTable (
Nom_de_colonne1 Type_de_donnée,
Nom_de_colonne2 Type_de_donnée,
...
)
AS SELECT NomChamp1,
NomChamp2,
...
FROM NomTable2
WHERE ...;
```

Тип данных

- Цифровой: SMALLINT (16 бит), Integer (32 бита), Float
- строки: CHAR (п) (фиксированная длина) VARCHAR (п)
- (Максимальная длина)
- Дата: DATE, TIME, TIMESTAMP

ограничения целостности

- назвать ограничение: CONSTRAINT
- установить значение по умолчанию: DEFAULT
- указать, что запись является обязательным: NOT NULL
- проверить, что введенное значение для поля уже не существует в таблице: UNIQUE
- Испытание на поле: CHECK

пример:

```
CREATE TABLE Clients(
Nom char(30) NOT NULL,
Prenom char(30) NOT NULL,
Age integer, check (Age < 100),
Email char(50) NOT NULL, check (Email LIKE "%@%")
)
```

Определение ключа

Напоминание: ключ является одним (или более) столбцов), чьи знания позволяет точно указать один кортеж.

- Первичный ключ: **PRIMARY KEY**

```
PRIMARY KEY (colonne1, colonne2, ...)
```

- Внешний ключ: **FOREIGN KEY...REFERENCES...**

```
FOREIGN KEY (colonne1, colonne2, ...)
REFERENCES
NomTableEtrangere(colonne1,colonne2,...)
```

Обновление информации

- добавить кортежи: **INSERT INTO**
  - вставить одну строку:

```
INSERT INTO NomTable(colonne1,colonne2,colonne3,...) VALUES
(Valeur1,Valeur2,Valeur3,...)
```

  - вставить несколько строк:

```
INSERT INTO NomTable(colonne1,colonne2,...) SELECT
colonne1,colonne2,... FROM NomTable2 WHERE qualification
```

(Неизвестные значения равны **NULL**)

- модифицировать существующие кортежей: **UPDATE...SET...WHERE...**

```
UPDATE NomTable SET Colonne = ValeurOuExpression
WHERE qualification
```

- удалить кортежи: **DELETE FROM...WHERE...**

```
DELETE FROM NomTable
WHERE qualification
```

манипулирование данными

Изменение таблицы

- удаление элементов: DROP (VIEW, INDEX, TABLE)

```
DROP TABLE NomTable
```

(Удаление данных и структуры таблицы)

- Удаление данных только: TRUNCATE

```
TRUNCATE TABLE NomTable
```

- переименовать таблицу: RENAME

```
RENAME TABLE AncienNom TO NouveauNom
```

- добавлять комментарии к таблице (или просмотра, или некоторые столбцы): COMMENT

```
COMMENT NomTable IS ’ Commentaires’ ;
COMMENT NomVue IS ’ Commentaires’ ;
COMMENT NomTable.NomColonne IS ’ Commentaires’ ;
```

изменения схем

С помощью ALTER вы можете изменить столбцы таблицы:

- Изменить типа столбца

```
ALTER TABLE NomTable
MODIFY NomColonne TypeDonnees
```

- добавлять новые столбцы

```
ALTER TABLE NomTable
ADD NomColonne TypeDonnees
```

- удалить столбцы

```
ALTER TABLE NomTable
DROP COLUMN NomCcolonne
```

(Возможно, если столбец не является частью зрения, не делает при условии ограничение целостности)

изменения схем (2)

- добавлять новые ограничения

```
ALTER TABLE NomTable
ADD CONSTRAINT NomContrainte
```

- удалять ограничения

```
ALTER TABLE NomTable
DROP CONSTRAINT NomContrainte
```

- включить или отключить все ограничения

```
ALTER TABLE NomTable
CHECK CONSTRAINT NomContrainte

ALTER TABLE NomTable
NOCHECK CONSTRAINT ALL
```

Создание представлений

Представление это виртуальная таблица, оценивается при каждом посещении (данные не хранятся в таблице в базе данных). Представление определяется близким SELECT.

- получить краткую информацию
- избегать раскрытия определенной информации
- обеспечить независимость внешней схемы

Создание представлений

- Синтаксис для определения представления является:

```
CREATE VIEW NomVue(NomColonne1,...)
AS SELECT NomColonne1,..
FROM NomTable
WHERE Condition
```

Exemple :

```
CREATE VIEW EtudiantsSrc(Nom,Prenom)
AS SELECT Nom,Prenom
FROM Etudiants
WHERE n_formation=12
```

Запросы к BD

Команда SELECT для запроса базы данных. Синтаксис:

```
SELECT [ALL|DISTINCT] NomColonne1,... | *
FROM NomTable1,...
WHERE Condition
```

- ALL: опция по умолчанию, выбирает все строки, которые удовлетворяют состояние
- DISTINCT: устраняет дубликаты

Предложение AS

Предложение AS позволяет поля в определенном запросе SELECT.

пример:

```
SELECT Compteur AS Ctp
FROM Vehicule
```

ограничение

Эти условия могут вызвать следующие операторы:

- Логические операторы: AND, OR, NOT
- цепи компараторов: IN, BETWEEN, LIKE
- арифметические операторы: +, −, /, %
- арифметическими компараторы: =, 6=, <, >, ≤, ≥, <>

пример:

```
SELECT *
FROM Vehicules
WHERE (Compteur>10000) AND (Compteur<=30000)
```

Ограничение на множестве

Предикаты BETWEEN и IN проверяют

- Что значение в пределах диапазона
- Что значение существует в списке значений

пример:

```
SELECT *
FROM Vehicules
WHERE Compteur BETWEEN 10000 AND 30000
SELECT *
FROM Vehicules
WHERE Marque IN ("Peugeot","Citroën")
```

Сортировка и другие операции

- сортировка: **ORDER BY ... DESC** или **ORDER BY ... ASC**

```
SELECT *
FROM Vehicules
ORDER BY Marque ASC, Compteur DESC
```

- средняя колонки: **AVG**
- количество строк в таблице: **COUNT**
- максимальное значение столбца: **MAX**
- минимальное значение столбца: **MIN**
- сумма значений в столбце: **SUM**

набор операций

Декартово произведение:

```
SELECT *
FROM NomTable1,NomTable2
WHERE ...
```

Эти две таблицы, на которых мы работаем, должны иметь тот же шаблон!

- Союз :UNION (держать дубликаты: UNION ALL)

```
SELECT Nom,Prenom
FROM Table_Employes
UNION
SELECT Nom,Prenom
FROM Table_Clients
```

- Пересечение: INTERSECT
- настроенная разность: MINUS

Спасибо за просмотр!