JOIN - HIERARCHIA

Jest to zwykły **JOIN,** gdzie tabela jest łączona sama ze sobą. Dzięki temu można odwzorować hierarchię pomiędzy rekordami w tabeli.

SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition;

INSERT INTO

Instrukcja **INSERT INTO** służy do wstawiania nowych rekordów w tabeli.

Możliwe jest napisanie instrukcji **INSERT INTO** na dwa sposoby.

INSERT INTO

Pierwszy sposób określa zarówno nazwy kolumn, jak i wartości, które należy wstawić:

INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

INSERT INTO

Jeśli dodasz wartości dla wszystkich kolumn w tabeli, nie musisz określać nazw kolumn w zapytaniu SQL. Upewnij się jednak, że kolejność wartości jest taka sama, jak kolumny w tabeli. Składnia INSERT INTO byłaby następująca:

INSERT INTO table_name
VALUES (value1, value2, value3, ...);

UPDATE

Instrukcja **UPDATE** jest używana do modyfikowania istniejących rekordów w tabeli.

UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;

UPDATE

Bądź ostrożny podczas aktualizowania rekordów w tabeli! **Zwróć uwagę na klauzulę WHERE w instrukcji UPDATE.** Klauzula WHERE określa, które rekordy powinny zostać zaktualizowane. Jeśli pominiesz klauzulę WHERE, wszystkie rekordy w tabeli zostaną zaktualizowane!

DELETE

Instrukcja **DELETE** służy do usuwania istniejących rekordów w tabeli.

DELETE FROM table_name
WHERE condition;

SELECT TOP/ LIMIT/ROWNUM

Klauzula **SELECT TOP** jest używana do określania liczby rekordów do zwrócenia.

Klauzula **SELECT TOP** jest użyteczna przy dużych tabelach, zawierających tysiące rekordów. Zwracanie dużej liczby rekordów może mieć wpływ na wydajność.

SELECT TOP/ LIMIT/ROWNUM

Nie wszystkie systemy baz danych obsługują klauzulę **SELECT TOP**. MySQL obsługuje klauzulę **LIMIT**, aby wybrać ograniczoną liczbę rekordów, podczas gdy Oracle używa **ROWNUM.**

SELECT column_name(s)
FROM table_name
WHERE condition
LIMIT number;

SELECT TOP/ LIMIT/ROWNUM

SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;

SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number;

UNION

Operator **UNION** służy do łączenia zestawu wyników dwóch lub więcej instrukcji SELECT.

- Każda instrukcja SELECT w ramach UNION musi mieć taką samą liczbę kolumn
- Kolumny muszą również zawierać podobne typy danych
- Kolumny w każdej instrukcji SELECT muszą być również w tej samej kolejności

UNION

```
SELECT column_name(s) FROM table1
UNION
SELECT column_name(s) FROM table2;
```

UNION ALL

Operator UNION domyślnie wybierze tylko różne wartości. Aby umożliwić powielanie wartości, użyj **UNION ALL**:

SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;

EXISTS

- Operator **EXISTS** służy do sprawdzania istnienia jakiegokolwiek rekordu w podzapytaniu.
- Operator **EXISTS** zwraca wartość true, jeśli podzapytanie zwraca jeden lub więcej rekordów.

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name
FROM table_name
WHERE condition);
```

INSERT INTO SELECT

Instrukcja **INSERT INTO SELECT** kopiuje dane z jednej tabeli i wstawia ją do innej tabeli.

INSERT INTO SELECT wymaga zgodności typów danych w tabelach źródłowych i docelowych.

Istniejące rekordy w tabeli docelowej nie są naruszone.

INSERT INTO SELECT

Polecenie SQL wstawiające wszystkie kolumny do nowej tabeli:

```
INSERT INTO table2
SELECT * FROM table1
WHERE condition;
```

INSERT INTO SELECT

Poniższe polecenie wstawia tylko niektóre kolumny z jednej tabeli do innej:

INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;

CREATE DATABASE

Instrukcja **CREATE DATABASE** jest używana do tworzenia nowej bazy danych SQL.

CREATE DATABASE databasename;

DROP DATABASE

Polecenie **DROP DATABASE** służy do usuwania istniejącej bazy danych SQL.

DROP DATABASE databasename;

CREATE TABLE

Instrukcja **CREATE TABLE** służy do tworzenia nowej tabeli w bazie danych.

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....);
```

CREATE TABLE

Kopia istniejącej tabeli może zostać utworzona przy użyciu kombinacji instrukcji **CREATE TABLE** i instrukcji **SELECT**.

```
CREATE TABLE new_table_name AS
    SELECT column1, column2,...
    FROM existing_table_name
    WHERE ....;
```

CREATE TABLE

Nowa tabela uzyskuje te same definicje kolumn. Można wybrać wszystkie kolumny lub konkretne kolumny.

Jeśli utworzysz nową tabelę przy użyciu istniejącej tabeli, nowa tabela zostanie wypełniona istniejącymi wartościami ze starej tabeli.

DROP TABLE

Instrukcja **DROP TABLE** jest używana do usuwania istniejącej tabeli w bazie danych.

DROP TABLE table_name;

TRUNCATE TABLE

Instrukcja **TRUNCATE TABLE** służy do usuwania danych wewnątrz tabeli, ale nie samej tabeli.

TRUNCATE TABLE table_name;

ALTER TABLE

Instrukcja **ALTER TABLE** służy do dodawania, usuwania lub modyfikowania kolumn w istniejącej tabeli.

Instrukcja **ALTER TABLE** służy również do dodawania i upuszczania różnych ograniczeń na istniejącej tabeli.

ALTER TABLE – ADD COLUMN

Aby dodać kolumnę do tabeli, użyj następującej składni:

ALTER TABLE table_name
ADD column_name datatype;

ALTER TABLE - DROP COLUMN

Aby usunąć kolumnę w tabeli, użyj następującej składni (zauważ, że niektóre systemy baz danych nie zezwalają na usuwanie kolumny):

ALTER TABLE table_name
DROP COLUMN column_name;

ALTER TABLE - ALTER/MODIFY COLUMN

Aby zmienić typ danych kolumny w tabeli, użyj następującej składni:

ALTER TABLE table_name
MODIFY COLUMN column_name datatype;

DROP COLUMN

Jeżeli chcemy usunąć kolumnę o nazwie "name" w tabeli "Persons" używamy następującej instrukcji SQL:

ALTER TABLE Persons
DROP COLUMN name;

CONSTRAINTS

Ograniczenia można określić, gdy tabela jest tworzona przy użyciu instrukcji CREATE TABLE lub po utworzeniu tabeli za pomocą instrukcji ALTER TABLE.

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
    column3 datatype constraint,
    ....
);
```

CONSTRAINTS

Ograniczenia są używane w celu zawężenia typu danych, które mogą zostać zapisane w tabeli. Zapewnia to dokładność i wiarygodność danych. Jeśli wystąpi jakiekolwiek naruszenie ograniczenia i działania danych, operacja zostanie przerwana.

NOT NULL

Domyślnie kolumna może zawierać wartości NULL. Ograniczenie NOT NULL powoduje, że kolumna NIE akceptuje wartości NULL.

Wymusza to fakt że pole zawsze zawiera wartość, co oznacza, że nie można wstawić nowego rekordu ani aktualizować rekordu bez dodawania wartości do tego pola.

NOT NULL

Następujący SQL zapewnia, że kolumny "ID", "LastName" i "FirstName" nie akceptują wartości NULL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

UNIQUE

Ograniczenie **UNIQUE** zapewnia, że wszystkie wartości w kolumnie są różne.

Zarówno ograniczenia UNIQUE oraz PRIMARY KEY gwarantują wyjątkowość kolumny lub zestawu kolumn.

Można jednak mieć wiele ograniczeń UNIQUE na tabeli, ale tylko jeden klucz główny na każdej z nich.

UNIQUE

Następujący SQL tworzy ograniczenie **UNIQUE** w kolumnie "ID", gdy tworzona jest osoba:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    UNIQUE (ID)
);
```

UNIQUE

Aby zdefiniować ograniczenie **UNIQUE** dla wielu kolumn, należy użyć następującej składni SQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

PRIMARY KEY

Ograniczenie **PRIMARY KEY** jednoznacznie identyfikuje każdy rekord w tabeli bazy danych.

Klucze podstawowe muszą zawierać wartości UNIQUE i nie mogą zawierać wartości NULL.

Tabela może zawierać tylko jeden klucz podstawowy, który może składać się z pojedynczych lub wielu pól.

PRIMARY KEY

Następujący SQL tworzy klucz główny w kolumnie "ID", gdy tworzona jest osoba:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
);
```

PRIMARY KEY

Aby stworzyć klucz główny oparty na wielu kolumnach, użyj następującej składni SQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

FOREIGN KEY

Klucz obcy jest kluczem służącym do łączenia dwóch tabel.

Klucz obcy w tabeli wskazuje na PRIMARY KEY w innej tabeli.

FOREIGN KEY

Następujący SQL tworzy klucz obcy w kolumnie "PersonID", gdy tworzona jest tabela "Orders":

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

INDEX

Instrukcja **CREATE INDEX** służy do tworzenia indeksów w tabelach.

Indeksy są używane do szybkiego pobierania danych z bazy danych. Użytkownicy nie widzą indeksów, są one po prostu wykorzystywane do przyspieszania wyszukiwania / zapytań.