Software Technologies for Data Science

Lecture 19

SQL Constraints and Keys

Ken Cameron

Content

- INSERT with SELECT
- SELECT expressions
- CONSTRAINTS
- KEYS

`INSERT` with `SELECT`

- Until now we've either used `INSERT` or `SELECT` in a query.
- It is possible to use both together.
  - This allows us to use data from one or more of the tables in the database
  - To add rows to another.
- Avoids copying the data to the application and then inserting new rows from the application.

INSERT with SELECT

```
INSERT INTO tablex (colx1, colx2, ..., colxn)
 SELECT coly1, coly2, ..., coly2 FROM tabley ...
```

- $\texttt{colx}_1, \texttt{colx}_2, \ldots, \texttt{colx}_n$ are the columns of `tablex`.
- $\texttt{coly}_1, \texttt{coly}_2, \ldots, \texttt{coly}_n$ are the columns produced by the select.

INSERT with SELECT

INSERT INTO tablex (**colx$_1$, colx$_2$, ..., colx$_n$**)
SELECT **coly$_1$, coly$_2$, ..., coly$_2$** FROM tabley ...

- colx$_1$, colx$_2$, ..., colx$_n$ are the columns of tablex.
- coly$_1$, coly$_2$, ..., coly$_n$ are the columns produced by the select.
  - Both lists must be the same length.

`SELECT` result

- Each result item in a SELECT is not restricted to a column name.
- It can be any expression.
  - And it can be as simple as a constant
  - Or as complicated as you need.
  - We've already seen the aggregate functions of `GROUP BY`.

`SELECT` expressions

- The SQL language supports all the usual operators.
  - Numeric, string, logical, etc.
- And a long list of useful functions.
  - Look them up.
- You can include any column from a table you have used in the `SELECT`.

Aggregate Functions

`BIT_AND()`	Bitwise and
`BIT_OR()`	Bitwise or
`BIT_XOR()`	Bitwise xor
`COUNT()`	Number of rows grouped
`COUNT(DISTINCT)`	Number of distinct rows grouped
`GROUP_CONCAT()`	Concatenated strings
`MAX()`	Maximum value
`MIN()`	Minimum value

Aggregate Functions

AVG()	The average value.
STDDEV()	The population standard deviation
STDDEV_POP()	The population standard deviation
STDDEV_SAMP()	The sample standard deviation
VARIANCE()	The population standard variance
VAR_POP()	The population standard variance
VAR_SAMP()	The sample standard variance

Constraints

- SQL allows us to place constraints on tables and columns.
  - This helps ensure the integrity of the data.
- We've already introduced a couple of them:
  - `NOT NULL` is a constraint on the contents of a column.
  - `PRIMARY KEY` is another constraint on the contents of a column.

Other Common Constraints

- `UNIQUE`
  - All entries must be unique.
  - This is also permits `NULL` as a value. But only one instance.
- `INDEX`
  - Indicates we wish to access the table contents via references to this column.
  - And therefore want it to be efficient.

Indexes

- Without an index, if SQL needs to find rows that match a value in a column it might search through the whole table.

- With an index, the search can be optimised.

  - For example, MySQL uses B-trees for most indexes.

  - It may also use R-trees (for spatial data) and hashing.

Other Common Constraints

- CHECK
  - Ensures that values satisfy a condition.
- DEFAULT
  - Used to set a default value for a column when no value is specified.
  - c.f. Setting the parameter in a python function definition.

Example

```
CREATE TABLE Persons (
    ID int NOT NULL,
    Name varchar(255) NOT NULL,
    Age int,
    Member int,
    City varchar(255),
    CHECK (Age >= 18),
    CONSTRAINT Chk_Person CHECK (Member=1 AND
                                 City='Bath')
)
```

FOREIGN KEY

- FOREIGN KEY is used to link tables together.
  - A child table contains a FOREIGN KEY.
  - That maps to a PRIMARY KEY in a parent or referenced table.
  - The two keys must be of the same type.
  - The constraint is that a value in the child table is only permitted if it exists in the parent table.

Example

- Consider the `studentid` from our common example.
  - In the students table it is the primary key.
- In all the other tables we should define it as a FOREIGN KEY.
  - And map it to the primary key.

Example

MySQL

```
CREATE TABLE enrolled (
        studentid INT,
        unitid INT,
        FOREIGN KEY (studentid) REFERENCES students(studentid),
        FOREIGN KEY (unitid) REFERENCES units(unitid),
)
```

SQL Server/Oracle/MS Access

```
CREATE TABLE enrolled (
        studentid INT FOREIGN KEY REFERENCES students(studentid),
        unitid INT FOREIGN KEY REFERENCES units(unitid)
)
```

Other Keys

- `PRIMARY, FOREIGN` and `UNIQUE` are not the only types of key.
- There are also
  - Candidate Keys
  - Alternate Keys
  - Super Keys
  - Composite/Compound Keys

Candidate Key

- A Candidate Key is a set of one or more fields/columns that

  can identify a record uniquely in a table.

- There can be multiple Candidate Keys in one table.

- Each Candidate Key can work as Primary Key.

Alternate Key

- An alternate key is one that could serve as a `PRIMARY KEY`.
  - But is n't currently the `PRIMARY KEY`.
- For example,

  In a table that uses the National Insurance Number as the `PRIMARY KEY`,

  The Unique Tax Reference could be suitable Alternate Key.

Composite Key

- Composite Key is a combination of

  more than one fields/columns of a table.

- It can be a Candidate Key

- And therefore, can be a Primary key

Super Key

- A Super Key is
  - A set of one or more keys
  - that can be used to identify a record uniquely in a table.
- Primary key, Unique key, Alternate key are subsets of Super Keys.

Why think about keys?

- Keys help us indentify the rows in a table we care about.
- Is every row in the table unique?
  - Do we need it to be?
- How complex does our `WHERE` clause need to be to identify the row(s) we're interested in?
  - Simpler will be faster.

Summary

- INSERT with SELECT
- SELECT expressions
- Constraints
- Keys