Lecture 16 – Constraints & Stored Procedures

CS2208
Information Storage and Management I

A TRADITION OF INDEPENDENT THINKING

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Dr Harry Nguyen
<hn@cs.ucc.ie>

What We Saw Last Lecture

- DDL
- Indexing
- Views

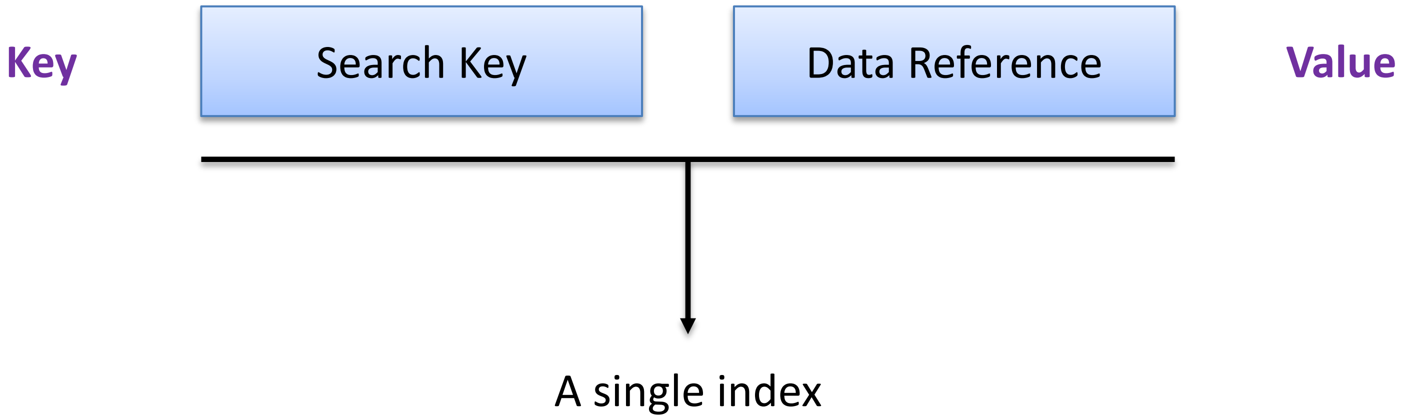
SQL Data Definition Language (DDL)

In the context of SQL, data definition or data description language (DDL) is a syntax for creating and modifying database objects such as tables, indices, etc.

INDEXING IN DBMS

Indexing is a technique to optimise operational time of database queries using data structures

Leads to faster query results, better database performance and lesser storage sizes.



Index Definition in SQL

- We can create an index on an attribute of a relation
  - This creates a data structure (e.g. B+ tree) that allow the system to efficiently search for tuples with a particular value for the attribute

- We can create an index using a syntax such as

```
CREATE INDEX <index_name> ON <relation>(<attributelist>)
```

where attributes list is the list of attributes that form the search key for the index

VIEWS

- Views are a mechanism that allow us to treat a query result as a relation
- They are useful for creating customized views of data related to some process
- They can also be used to restrict the views of full relations for some users
- For example, we might create the following view for an office clerk that hides personal information about instructors (e.g., salary)

Creating Views

- We can create views as follows:

```
CREATE view V AS <query expression>;
```

- e.g.,

```
CREATE VIEW faculty AS
SELECT id, name, dept_name
FROM instructor;
```

- We can then run queries against the view as would against a relation:

```
SELECT * FROM faculty;
```

Integrity Constraints – What are they?

CS2208
Information Storage and Management I

Integrity constraints ensure that changes made to databases do not cause any inconsistencies

These constraints are defined when a table is created

We can either write SQL to capture these constraints or we can use graphical interfaces in WorkBench

We can also add constraints using an alter statement but we should be careful if the table is populated with existing data

Not Null

Any attribute in a relation can have a null value – null is a member of all domains

In some cases, null is an inappropriate value

- We can make a variable not null as follows:

  name varchar(20) not null

- For example, an ID name or name of student or employee would not be useful if they had a null value
- We can declare a value to be *not null* that is, must have a value
- It is a domain constraint – it affects how we interact with a domain
- primary keys must be *not null*

UNIQUE

- Unique means that for some attribute, no two attribute values in the list of all attribute values can be the same
  - That is they must be unique
- Unique variables can be null however

CHECK

- Check allows us to ensure that attribute values satisfy specific conditions
- For example, that a salary should be greater than zero

```
CREATE TABLE instructor
(
   id int,
   name varchar(20) NOT NULL,
   dept varchar(20),
   salary numeric(8,2) CHECK (salary > 0),
   PRIMARY KEY (id)
);
```

REFERENTIAL CONSTRAINTS

- We often wish to ensure that a value that appears in one relation also appears in some other relation.
  - For example, department id for instructor
  - By adding a foreign key constraint we ensure that the referenced department must already exist.
  - The referenced attribute must be a primary key or a unique value and must be compatible with the attribute described in the referencing table.

```
CREATE TABLE instructor
(
    id int,
    name varchar(20) NOT NULL,
    dept varchar(20),
    salary numeric(8,2) CHECK (salary > 0),
    PRIMARY KEY (id),
    FOREIGN KEY (dept_name) REFERENCES department
);
```

Cascades

- If a referential integrity constraint is violated, the attempted operation is rejected
  - For example, if we try to delete a row in a table that is also referred to through a foreign key – for example, a department referenced by a instructor record – then the operation would be rejected.

- We can override this and allow a delete of a department to also delete any referring instructors

FOREIGN KEY (deptid) references department on delete cascade

- We can keep the instructors and replace the department id with null as follows:

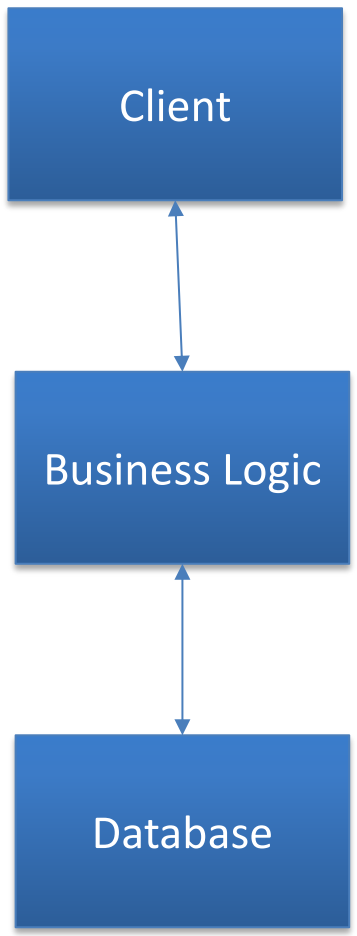
FOREIGN KEY (deptid) references department on set null cascade

- We can declare these cascades for delete or update operations

LIMITATIONS OF SQL

- SQL is generally a declarative language
  - We specify the information that we need or want to modify but we leave it to the DBMS to make the modifications
- Procedural languages, such as Python, Java, C/C++, etc allow us to specify a sequence of actions and how they will occur:
  - We sometimes need this degree of control to express processes that are used by an enterprise.
- As a result, we often see software architectures that are three or n tier – with the database at the lowest tier.

TYPICAL THREE TIER ARCHITECTURE



- User Interfaces

- Business Logic

- Data management

Databases and Software Development

- Typically we would use a library or framework to manipulate the database
  - JDBC in Java
  - ODBC in Visual C++ and Visual Basic
  - PyMySQL in Python

- Manipulation of the database is often driven from the code:
  - Some frameworks are explicitly organized that way, for example Django
  - In other cases, there are several arcs of software development, Webapp, iOS app, Android app, perhaps all communicating with the database directly.
    - In this case, the architecture can become fragmented and result in duplication of typical interactions with the database

Stored Procedures and Functions

- Most DBMS allow us to define both stored procedures and functions
- Each DBMS has its own syntax for doing this
- We can embed common actions in our database into SQL using functions
  - We've already used functions, such as SUM(), AVG(), etc
- Stored Procedures capture processes that we might wish to implement
  - We can call these from our SQL or we can invoke them through a library call from a procedural language

Stored Procedure - Example

```
CREATE PROCEDURE `book_room`(
  IN roomid int,
  IN booker varchar(45),
  IN roomname varchar(45),
  IN guestnumber int,
  IN eventdate varchar(45),
  OUT result int)

BEGIN

  DECLARE roomlimit int;
  SELECT roombookinglimit.limit into roomlimit FROM roombookinglimit WHERE
roombookinglimit.roomname = roomname;

  IF roomlimit >= guestnumber THEN
    INSERT INTO roombooking values (roomid, booker, roomname, guestnumber,
eventdate);
    SET result = 0;
  ELSE
    SET result = 1;
  END IF;

END
```

SUMMARY

Integrity Constraints

Database and Software Development

Stored Procedures