SQL Programming

Intro

©1998-2018 / Bergin

SQL programming involves the
design, coding, and testing
of SQL programs, for the purpose of
managing *information structures*
within a database.

©1998-2018 / Bergin

Page A-2: Information Structures

The information structures of a database include:

TABLES that hold data;

VIEWS that provide dynamic snapshots of table data;

Internal Structures of the database that define who owns what data, as well as who is permitted to use that data.

Database systems include other structures than the few I have listed here, but this is all you need to concern yourself with, at this stage.

©1998-2018 / Bergin

Page A-3: Table Genesis

Tables are the most common information structure in a database, and this begs the question:

Where do they come from?

All well-managed Information Technology projects follow a methodology known as the Systems Development Life Cycle (SDLC). This SDLC provides a road map for the IT professionals to follow as they work from the inception of a project, thru to its deployment and maintenance.

©1998-2018 / Bergin

Page A-4: SDLC

The generic SDLC model is comprised of these four stages:

- Analysis
- Design
- Implementation
- Maintenance and Support

During the Analysis stage, the systems analyst, working with the user community, prepares a rough sketch of the application requirements as identified by the client.

This requirements document is transformed into a rigorous set of specifications that thoroughly describe every element of the application. Think of these specification as being analogous to a set of blueprints for a home. Some of the elements that are defined in this blueprint include:

- data structures;
- program requirements; and
- training requirements.

©1998-2018 / Bergin

Page A-5: SDLC - Implementation

During the implementation stage, the system is 'put together': some elements may be purchased, other elements may be developed, and it is during this stage where the programming work for an application is typically done. By this stage of the project, a significant amount of work has already been completed and we haven't even written any code!

Maintenance and Support is that stage of the life cycle where computing resources (hardware, software, and people) are managed to keep the application up-to-date with changes in the business, and to insure that day-to-day operations continue to run smoothly.

©1998-2018 / Bergin

Page A-6: Table Genesis - Revisited

We are now better equipped to answer the question posed earlier:

Where do tables come from?

The need for specific data elements is identified during the analysis stage. Those needs are transformed into data specifications that mandate the use of database tables that will serve as the repositories of this data.

The tables themselves are built during the early part of the implementation stage, when they are also initialized with data.

So, by the time we get around to writing any programs, the initial structure of the database has been prepared and loaded with data.

©1998-2018 / Bergin

Page A-6: Contrivance

Now having said all of that, recognize that you don't need to understand every nuance of the SDLC in order to master the SQL programming language.

These lessons and activities have been designed to help you develop your SQL programming skills without getting bogged down in all of the surrounding details of applications development.

There will be times when you'll mutter to yourself "This has nothing to do with the real world! I'll never be asked to write a program to do this!!!"

And indeed, your intuitions will be 'right on'. And when these insights do occur, please reflect on why you're here:

to learn

and master

the features of the SQL programming language. Mastery requires practice and experimentation.

©1998-2018 / Bergin

Page B-1: Managing Information Structures

Prior to database technology, institutional data was managed under the 'file processing' paradigm.

Each application might require a number of files of data, and these files would be managed independently of one another. Programmers would write utility programs to create and destroy these files, another set of utility programs to add and delete records from these files, along with a number of programs that could extract needed data from these files.

Finally there would be a series of programs that would implement the business policy that the application was intended to support.

Consider a student record system that might be required to store information about:

- students
- courses
- faculty
- rooms
- transcript

©1998-2018 / Bergin

Page B-2: Student Record System

In a traditional file processing system there would be 5 separate utility programs to create each of these files.

Another five programs to add and/or delete records to these files. Another series of programs to extract data from each of these files, …. You get the idea.

All of this housekeeping had to be done for each and every file in the system. This was rather expensive overhead when you stop to think that what the application system was designed for was to support the business needs.

Lots of time on file maintenance, not so much time on implementing the business rules.

©1998-2018 / Bergin

And then one day someone had the bright idea to design a tool that provided for the commonality of these functions.

That tool was a DBMS and it was designed to increase the productivity of the programmer by taking care of these housekeeping details for them. DBMS's streamlined the work that was required to handle the CRUD.

CRUD is an acronym that stands for:

create
retrieve
update
delete

CREATE is concerned with adding new records

RETRIEVE extracts records that satisfy certain criteria

UPDATE is used to change or update the values of fields

DELETE is concerned with removing or deleting records

©1998-2018 / Bergin

Page B-4: SQL and CRUD

As you would expect, there is a direct correlation between SQL and the functionality that it provides in support of CRUD.

Separate SQL statements are used for each of these functions:

CRUD	SQL
create	insert
retrieve	select
update	update
delete	delete

There are also special SQL statements to handle the creation and deletion of files (tables), as well as special statements to permit different groups of users to perform different CRUD actions on the different database structures.

In this last instance, for example, some users may be allowed to retrieve information from the STUDENT table, while being disallowed to update those same records.

©1998-2018 / Bergin

Page B-5: Classification - Traditional

The SQL language includes more than just these four statements, and it might be helpful if we take a moment now to look at ways of classifying and categorizing the different statement types.

The traditional scheme for classifying SQL statements used only three categories:

data manipulation language (DML) commands;

data definition language (DDL) commands; and,

data control language commands (DCL).

DML includes the commands that support CRUD: INSERT, SELECT, UPDATE, and DELETE.

DDL includes the commands that create (and remove) the database structures: CREATE and DROP.

DCL includes the commands that assign permissions to the database objects: GRANT and REVOKE

©1998-2018 / Bergin

Page B-6: Classification – 1999 Standard

The 1999 standard for ANSI SQL proposes these categories for the language elements:

Data Statements: SELECT, INSERT, UPDATE, DELETE

Schema Statements: CREATE, DROP, ALTER

Transaction Statements: COMMIT, ROLLBACK

Session statements: SET

Connection Statements: CONNECT, DISCONNECT

©1998-2018 / Bergin

Page B-7: Classification – Oracle

And the Oracle database company proposes yet another kind of classification scheme. Here's the categories they highlight (taken from product documentation for the 9i release of their database product):

DML: SELECT, UPDATE, INSERT, and DELETE;

DDL: CREATE, DROP, and ALTER;

Transaction control statements:COMMIT. ROLLBACK, and SAVEPOINT;

Session control statements: ALTER SESSION, and SET ROLE;

System control statements: ALTER SYSTEM;

Embedded SQL statements: OPEN, CLOSE, FETCH, and EXECUTE.

©1998-2018 / Bergin

Page B-8: Classification – Bil

Here is the scheme we'll be using:

DQL – data query language – SELECT;
DML – data manipulation language – INSERT, DELETE, and UPDATE;
DDL – data definition language – CREATE and DROP; and,
DCL – data control language – GRANT and REVOKE.

As far as those other schemes are concerned, you should note them, and anticipate that our model will be refined as we learn more about SQL, especially as we advance to the Level 2 course. But here in Level 1, those other schemes just don't emphasize the essential elements of the language.

The classification scheme represents the core set of statements needed to manage the information structures of any database.

©1998-2018 / Bergin

Page C-1: Meta Data

When you stop to think about it, a database is a rather complex application, so complex in fact, that it could probably use a database just to manage its internal structures.

And this brings us to one of the really *COOL* features of modern databases. They are self-describing! They use their own structures to define themselves. It kind of makes sense – I mean, why go to all the trouble to build a separate database to manage the database, when you ought to be able to use the database itself, to manage things.

If this wordplay is too confusing for you, then try this.

In addition to storing the data of the business application, databases also store metadata (ie data or information about the data elements) to help it manage the database.

©1998-2018 / Bergin

Page C-2: Meta as a prefix

As a brief digression, meta is an interesting prefix in the English language. Consider these definitions of words that include meta as a prefix:

Metadata: data abut data

Metalanguage: a language about languages, or a language used to discuss other languages, the language of languages.

Metatheory; a theory of theories, a theory used to describe other theories.

©1998-2018 / Bergin

Page C-3: Database Definitions

This brings us to a couple of better working definitions for a database:

A database is the collection of the occurrences of multiple record types, containing the relationships between records, data aggregates, and data items.(Martin, 1975).

The phrase: occurrences of multiple record type implies that the database contains more than a single file's worth of data, in fact, a DBMS derives most of its value by serving as an integrated data repository for the business.

And not only does it contain the business data, (for example STUDENT and FACULTY data), but it also includes the relationships among these records. For example: the relationship that faculty TEACH students, or that faculty ADVISE students, or that students EARN GRADES FROM faculty.

©1998-2018 / Bergin

Page C-3: Definitions (cont.)

Database: *a collection of logically related data that supports shared access by many users and is protected and managed to retain its value.* (Loomis, 1987)

The author reinforces the notion of logically related data, and introduces the notion that the data is shared. This is critical! Databases serve as an integrated repository of data for the enterprise, hence they are used by more than a single user, or sngle department, or even a single division. They are a shared resource.

And once you gather all of your eggs into one basket, you best guard that basket well. Information is one of the most important and valuable corporate resources, and it must be protected and managed (lest it lose its value).

©1998-2018 / Bergin

Page C-4: Definitions (cont.)

A database is a self-describing collection of integrated records (Kroenke, 2003).

That pretty much says it all – I hope you appreciate its full meaning.

What does it mean to be self-describing?

What does it mean to be a collection of integrated records?

©1998-2018 / Bergin

And the point of all of this wordplay?

To understand the scope and depth of SQL you need to have an appreciation of databases. Anything that needs to be done in the database must be accommodated by SQL.

Every database action has a SQL command.

©1998-2018 / Bergin

Page D-1: SQL Examples

Remember that our classification scheme for the SQL commands includes these categories:

DQL – data query language – SELECT;

DML – data manipulation language – INSERT, DELETE, and UPDATE;

DDL – data definition language – CREATE and DROP; and,

DCL – data control language – GRANT and REVOKE.

Here are examples of commands from each of these categories.

©1998-2018 / Bergin

Page D-2: DQL - SELECT

DQL: Data Query Language

Purpose: To retrieve, or read, records from the tables in the database. DQL supports the R in cRud.

Statement: SELECT

Example

```
SELECT  last_name
FROM    talent;
```

©1998-2018 / Bergin

Page D-3: DML - INSERT

DML: Data Manipulation Language

Purpose: To modify or manipulate the (business) data that is stored in the database. DML supports the C-U-D of CRUD.

Statement: CREATE

adds/inserts a record

Example

```
INSERT
INTO   talent(last_name,first_name)
       VALUES("Roberts", "Julia");
```

©1998-2018 / Bergin

Page D-4: DML - DELETE

DML: Data Manipulation Language

Purpose: To modify or manipulate the (business) data that is stored in the database. DML supports the C-U-D of CRUD.

Statement: DELETE

deletes/removes a record

Example

```
DELETE
FROM   talent
WHERE (last_name = "Roberts");
```

©1998-2018 / Bergin

Page D-5: DML - UPDATE

DML: Data Manipulation Language

Purpose: To modify or manipulate the (business) data that is stored in the database. DML supports the C-U-D of CRUD.

Statement: UPDATE

updates/modifies a record

Example

```
UPDATE talent
SET first_name = "Ron"
WHERE (last_name = "Roberts");
```

©1998-2018 / Bergin

Page D-6: DDL - CREATE

DML: Data Definition Language

Purpose: To add, delete. or modify one of the structures in the database. DDL does not support CRUD functions – because CRUD functions pertain to the business data elements and not the 'containers' that hold those elements

Statement: CREATE
adds a structure

Example

```
CREATE VIEW  actors
AS
        SELECT *
        FROM talent;
```

©1998-2018 / Bergin

Page D-7: DDL - DROP

DML: Data Definition Language

Purpose: To add, delete. or modify one of the structures in the database. DDL does not support CRUD functions – because CRUD functions pertain to the business data elements and not the 'containers' that hold those elements

Statement: DROP

removes/deletes a structure

Example

```
DROP VIEW  actors;
```

©1998-2018 / Bergin

Page D-8: DCL - GRANT

DML: Data Control Language

Purpose: To help control access to the data in the database.

Statement: GRANT
bestows a permission

Example

```
GRANT  select ON talent
TO   public;
```

©1998-2018 / Bergin

Page D-9: DCL - REVOKE

DML: Data Control Language

Purpose: To help control access to the data in the database.

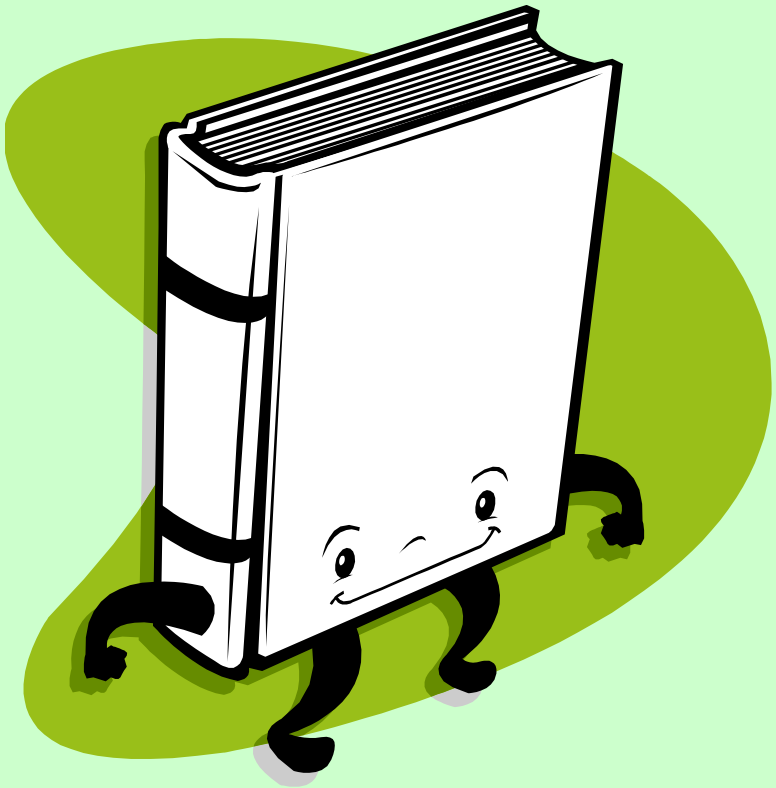
Statement: REVOKE
withdraws a permission

Example

```
REVOKE  select ON talent
FROM  public;
```

©1998-2018 / Bergin

Page T-1: Terminology



CRUD: create, retrieve (read), update, delete

Meta: metadata, metalanguage, metatheory

Self-describing

integrated

DQL, DML, DDL, DCL

©1998-2018 / Bergin

Page Z-1: End Notes

Many thanks to Caitlin Bergin who designed and implemented a number of the sample tables that we'll be using in this course.

Please drop me an email if you noticed any errors in this module. I'd also appreciate reading your comments, criticisms, and or suggestions as to how this module could be improved.

Thanks,

bil

©1998-2018 / Bergin

That’s All

©1998-2018 / Bergin