Next Generation Higher National Unit Specification

Database Design and Development (SCQF level 8)

Unit code: J7DV 48

SCQF level: 8 (24 SCQF credit points)

Valid from: session 2023–24

Prototype unit specification for use in pilot delivery only (version 1.0) June 2023

This unit specification provides detailed information about the unit to ensure consistent and transparent assessment year on year.

This unit specification is for teachers and lecturers and contains all the mandatory information required to deliver and assess the unit.

This edition: June 2023 (version 1.0)

Unit purpose

This unit provides learners with the knowledge and skills they need to design a relational database and access and analyse its data. It also gives learners important knowledge and skills in the development and data analysis of NoSQL databases. This is a specialist unit, aimed at learners with an interest in database design and data analysis. It is particularly suitable for learners who are studying software development or data science.

The unit does not require any previous knowledge of database design or development. However, learners should have a basic understanding of what a database is and why it is essential to a business or organisation.

Completing the unit gives learners important knowledge and skills in the design, development and data analysis of relational and NoSQL databases.

Learners can progress from the unit to higher level (SCQF level 8 or above) content in database application development or to specialised vendor certifications in SQL.

Unit outcomes

Learners who complete this unit can:

1 produce a relational database design
2 implement a relational database design
3 use SQL to manage database tables and objects
4 use SQL to access data and perform transaction processing
5 create user-defined functions in SQL
6 use SQL to manage user accounts
7 explain NoSQL database principles
8 manage the data in a NoSQL database
9 analyse the data in a NoSQL database

Evidence requirements

This unit is assessed by product evidence. Knowledge evidence is inferred from the product evidence.

The product evidence consists of:

- a normalised relational database, appropriately populated, from a given specification
- a set of queries that learners use to access, analyse and manage the data
- a NoSQL database, appropriately populated
- a set of queries that learners use to manage and analyse the NoSQL data

Learners must design and populate the relational database. They must:

- create an entity-relationship diagram
- perform data normalisation
- verify that both methods produce tables in third normal form
- use SQL to join tables with appropriate key fields and uniqueness constraints
- ensure appropriate referential integrity is included, as well as measures to ensure data correctness
- add new data to an existing table and add data from an existing source

The evidence for the set of queries to access, analyse and manage the data must include:

- basic and advanced filter techniques
- sorting and sub-sorting
- grouping and sub-grouping with filtering
- calculated fields
- single-row and aggregated data functions
- multiple tables joined together appropriately

- correlated and non-correlated sub-queries
- filtering with regular expressions
- grouping of multiple statements into a transaction
- creation of user-defined functions to perform custom calculations on data
- multiple user accounts, created with different permissions on different database objects
- creation and rollback of a transaction upon failure

The evidence for the NoSQL database must include:

- data extracted from the relational database and stored in a non-relational NoSQL database
- a set of queries that retrieve, update and delete data from the NoSQL database

Learners' product evidence must be produced without assistance. The evidence can be produced over an extended period under lightly-controlled conditions. Learners should have access to learning materials. Learners must authenticate their evidence to validate the integrity of their submission. For example, learners should include their initials to all stored database objects.

Knowledge and skills

The following table shows the knowledge and skills covered by the unit outcomes:

Knowledge	Skills
Learners should understand: ♦ data modelling ♦ data normalisation ♦ data modelling and normalisation integration ♦ the data dictionary ♦ database table creation and management ♦ database data creation and management ♦ referential integrity creation and management ♦ check constraint creation and management ♦ database constraint creation and management ♦ database view creation and management ♦ database sequence creation and management ♦ database index creation and management ♦ user account creation and management ♦ data analysis features of SQL ♦ database transactions creation and management ♦ user-defined functions ♦ NoSQL database principles ♦ NoSQL database data management and analysis	Learners can: ♦ construct an entity-relationship diagram (ERD) ♦ construct an enhanced ERD (EERD) ♦ construct a relational schema from a given ERD or EERD ♦ apply normalisation techniques [first normal form (1NF), 2NF and 3NF] to an existing dataset ♦ cross-check a relational schema to verify data is normalised to 3NF ♦ create tables with appropriate data types, primary keys and uniqueness constraints ♦ create tables in a database from an existing table structure ♦ add, delete or update table columns ♦ delete tables from the database ♦ rename tables in the database ♦ implement referential integrity ♦ implement check constraints to both tables and table columns ♦ add, remove, enable and disable constraints ♦ implement simple and complex data views ♦ update or delete a view from the database ♦ generate unique identifiers for records ♦ implement and manage database indexing ♦ write queries to locate table structural information from the data dictionary ♦ add new data into an existing table ♦ insert data from an existing table into one or more tables

Knowledge	Skills
	Learners can: ♦ modify existing data in a table ♦ combine data from multiple tables ♦ write user-defined functions ♦ test and, where required, debug user-defined functions ♦ perform queries using the SQL SELECT statement ♦ limit the rows retrieved by a query using basic and advanced filter techniques ♦ sort data in either ascending or descending order ♦ perform arithmetic, date and time calculations in queries ♦ perform string manipulations in queries ♦ format output using time, date and number formats ♦ perform SELECT queries containing conditional constructs ♦ join multiple tables together using equi-joins, non equi-joins and cross-joins ♦ query multiple tables joined together using self and outer joins ♦ perform SELECT queries that return aggregated data in groups and sub-groups ♦ perform SELECT queries that return filtered data in groups and sub-groups ♦ perform SELECT queries that return data from queries linked together using common set operators ♦ perform SELECT queries that return data from queries linked together using basic and correlated sub-queries ♦ filter query output using regular expression meta-characters ♦ implement and manage user accounts ♦ assign and manage appropriate permissions to control access to database resources

Knowledge	Skills
	Learners can: ♦ implement transactions processing ♦ apply NoSQL database principles ♦ create a NoSQL database ♦ store, retrieve and manipulate data in a NoSQL database

Meta-skills

Throughout this unit, learners develop meta-skills to enhance their employability in the computing and information technology sector.

Self-management

This meta-skill includes:

- focusing: exploring the different approaches and methods of performing tasks, such as filtering data
- adapting: awareness that there are often several ways to solve a problem
- initiative: critically reviewing own database designs to increase efficiency

Social intelligence

This meta-skill includes:

- communicating: conveying technical information to different audiences
- collaborating: working in groups to discuss, analyse and formulate a solution to a given problem

Innovation

This meta-skill includes:

- curiosity: exploring the additional features and functions available in SQL
- sense-making: appreciating the power of SQL as a programming language

Delivery of unit

This unit provides learners with an understanding of how databases are designed. They also learn about tools to use to access and analyse their data. You should give learners several problems of each type to solve so that they gain a good understanding of the principles involved.

We suggest the following distribution of time:

Outcome 1 — Produce a relational database design
(15 hours)

Outcome 2 — Implement a relational database design
(10 hours)

Outcome 3 — Use SQL to manage database tables and objects
(20 hours)

Outcome 4 — Use SQL to access data and perform transaction processing
(30 hrs)

Outcome 5 — Create user-defined functions in SQL
(15 hrs)

Outcome 6 — Use SQL to manage user accounts
(10 hours)

Outcome 7 — Explain NoSQL database principles
(5 hours)

Outcome 8 — Manage the data in a NoSQL database
(10 hrs)

Outcome 9 — Analyse the data in a NoSQL database
(5 hours)

Professional recognition

Although not geared towards any specific vendor exam, this unit presents materials that are essential for learners to efficiently use any relational database platform.

Learners could progress from the unit to more specialised vendor certifications in SQL, such as the Oracle Database SQL Certified Associate (1Z0-071).

Additional guidance

The guidance in this section is not mandatory.

Content and context for this unit

Centres should create opportunities for learners to experiment in a safe environment and for independent learning and assessment. There are no restrictions on the database platform learners use. A possible platform is Oracle's free online APEX environment. Other cloud service providers provide similar database platforms that learners can use. Several NoSQL databases are available through Amazon Web Services, including DynamoDB and MongoDB.

We encourage centres to allow learners to appreciate the database-neutral nature of the unit so that they can find out how the same results can be achieved on different platforms.

Produce and implement a relational database design

Learners begin to understand that databases store data in tables, and that they exist to provide information obtained through queries on the database. Introduce learners to data modelling and the ideas of entities and attributes and their associated properties.

Demonstrate to learners how to construct ERDs. Introduce them to the one-to-one, one-to-many and many-to-many relationship cardinalities, as well as the idea of optional and mandatory cardinality constraints, and uniqueness constraints. Learners construct various ERDs using these basic principles. Introduce an appropriate ERD notation at this point. Although several are available, a very common one is the Barker notation, but alternatives are also acceptable.

Learners build on the basic ERD principles to learn to construct more enhanced ERDs (EERDs). These include the principles of entity supertypes and subtypes, such as:

- the exhaustive and mutually exclusive rules
- the concept of relationship transferability
- the concept of hierarchical and recursive relationships
- the method for resolving a many-to-many relationship into a pair of one-to-many relationships

Make learners aware of the exclusive-or constraint and how to apply it to EERDs.

Instruct learners how to transform an enhanced entity-relationship diagram into a relational schema. You should cover mappings for each of the earlier ERDs and EERDs, as well as the choosing of primary and foreign keys. These could be single, composite or artificial as required.

You should make learners aware that the 'bottom-up' approach of normalisation can be used in addition to the 'top-down' ERD database design technique. Learners should understand that normalisation, while an alternative approach to database design, is best used to verify the correctness of the relational schema produced by an ERD to relational schema mapping.

You should first explain to learners about the dangers of storing data in an un-normalised form. They should understand the issues surrounding insertion, update, and deletion anomalies.

Teach learners how to identify the functional dependencies in data, then take the existing data required to be stored in a relational database and normalise it through its first, second and third normal forms. You can ignore higher normal forms, as, at this level, learners are unlikely to encounter these in practice.

To allow learners to see how a correct EERD relational schema mapping produces tables in the third normal form, you should use the normalisation rules to verify a previous mapping is indeed correctly normalised.

Use SQL to access data

From this point onwards, learners require access to a relational database resource. Although several online and cloud databases are available, an excellent free resource is the Oracle Apex program. Learners can use this at Oracle APEX. Cloud solutions may be required for user account management.

Introduce learners to the structured query language (SQL) as a language used to access various relational database platforms. As part of this introduction, you should define and give an overview of data manipulation language (DML); data definition language (DDL); data control language (DCL); and transaction control language (TCL). Do not go into specific details, as this is intended to make learners aware of the scope of SQL.

As a starting point to using SQL, learners construct SELECT statements to extract all or specific columns of data FROM a single database table. Then teach them how to apply the WHERE clause to restrict the rows returned. These filter clauses should include equality, inequality and testing for NULL data, as well as string-matching using LIKE.

During database design, learners know that calculated fields are not included in database tables. At this point, demonstrate how to add calculations into SELECT statements to perform basic arithmetic on data. Introduce column aliases to make the output presentable. Learners should also be capable of formatting output using string concatenation.

You should now introduce more advanced filtering clauses. These include the ability to specify a range of values to be returned, as well as values matching multiple fields. You should include Boolean operators AND, OR and NOT. Demonstrate partial string-matching techniques, for example all records starting with the letter ‘A’ being returned.

You then show learners how to sort query output, using the ORDER BY clause. This should be by single columns as well as sub-sorts on columns in columns. Learners should be able to sort data in both ascending and descending order.

To allow learners to appreciate the range of built-in functions available in SQL, show them how to use common single-row string manipulation functions that manipulate selected data. Teach learners how they can use these in filters and to format output. Common examples must include: CONCAT, SUBSTR, LENGTH, INSTR, LPAD, RPAD, TRIM, and REPLACE. Demonstrate to learners how to use ROUND, TRUNC and MOD to alter numbers and

perform calculations on dates using ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN, NEXT_DAY, and SYSDATE. Learners should also format date and number output in common formats such as dd/mm/yy, 1,000,000.56, for example. You should also teach learners how to provide alternative output using CASE expressions.

Demonstrate to learners how to join two or more tables together and extract information from them. We expect them to understand the effect of performing various joins, including natural-joins, cross-joins, left, right and full outer joins, self-joins and non equi-joins.

Learners should know how to aggregate data using the GROUP BY clause and filter the summarised data using the HAVING clause. Instruct them on how to use common numerical grouping functions such as COUNT, MIN, MAX, SUM and AVG. Show them how to apply these calculations to unique data using DISTINCT. Finally, they should be capable of summarising sub-groups in other groups.

The functions listed above are available on an Oracle platform, but equivalent functions from other platforms are acceptable alternatives.

Learners should also link queries together using the common set operators: UNION, UNION ALL, INTERSECT and MINUS.

You should then instruct learners where to use basic sub-queries and correlated sub-queries. Demonstrate how to link queries together using each of these methods to display desired output. Also, learners should filter by the result of the sub-query — you should include single-row or multiple rows. Explain the use of operators such as IN, ANY, ALL, EXIST, and NOT EXIST.

Demonstrate to learners how to use regular expressions to perform more complex pattern matching in query filters. They should understand how to use common meta-characters such as *, |, ^, $, [], {m}, {m,n}, +, ?, ., (), as well as specific character types.

Use SQL to manage database tables

Learners should create, update and remove database tables.

You should make learners aware of the data dictionary in a database and explain how it contains a list of all its objects, including tables and their structure. You should then show learners how to locate their own objects using the DESC statement and by running a SELECT statement against the appropriate data dictionary view.

You should then demonstrate to learners how to create a new table from scratch. Introduce them to the various data types supported by their database platform and instruct them how to create a table using them correctly. The table creation should include default values, primary keys and uniqueness constraints.

Learners should also be capable of creating a new table from existing data stored in a separate table.

You should then show learners how to populate existing tables with data using INSERT INTO. This data should include timestamp and user information. The data source can be specified inside INSERT INTO, but it may also be from an existing table.

You should then show learners how to update and delete some or all data in a table.

Teach learners how to merge data from multiple tables into a single table and how to insert data into multiple tables from a single table. Then demonstrate how to alter an existing table. Learners should be capable of using the ALTER TABLE command to add, change and delete columns in a table, as well as alter its default values. Make learners aware that their database platform imposes limits on what alterations they can actually do. They should also be capable of removing tables in a database.

Learners should implement NOT NULL, UNIQUE, PRIMARY KEY, CHECK and FOREIGN KEY database constraints to their database table. Learners should implement primary key constraints to a single column in a table, as well as designate non-key columns as UNIQUE or NOT NULL. They should also implement the primary key at the table level if multiple columns are involved. To facilitate the proper joining of tables, learners should understand the principles of referential integrity and why we consider it good practice. They should then implement it as part of a foreign key constraint. Finally, teach them about the various CASCADE options that are available.

Use SQL to manage database objects

Learners should implement check constraints both at column and table levels to ensure data integrity is maintained.

Learners should implement constraint management using the ALTER TABLE command. They should be capable of adding, removing, enabling, disabling and dropping constraints from a table as required.

Introduce learners to the idea of a database view and explain why they are an important security consideration. Learners should be able to create a view from one or more tables and query it to verify its content. To make the view more user-friendly, teach learners that they can use column aliases. Teach learners when it is and isn’t permissible to update data through a view.

Learners should then appreciate the differences between simple and complex views. Teach them about the additional restrictions complex views place on users who need to update data through the view.

Learners should be able to update and remove a view from the database.

Demonstrate to learners how to generate sequential numbers that are applied automatically to each new record added to a table using sequence objects. They should be able to create, alter and remove them.

Explain how database indexes can hasten query execution. Define the difference between unique and non-unique indexes, and how to apply the index to a table column. They should also know how to drop an index.

Create user-defined functions

Demonstrate to learners the basics of how to create their own functions for analysing data in a table. These should be carefully tested using a correct range of test data. Learners create functions that can take zero or more arguments and return values. Explain how the function uses variables, loops and decisions to calculate its result.

Limit the content of the user-defined functions to basic tasks. Perhaps expand on the common functions already available. Examples include password length and content verification, user ID creation, and simple calculations.

If an Oracle platform is used, this should be PL/SQL, however any database-specific language is acceptable.

Use SQL to manage user accounts

Teach learners about user account creation. They should be able to create, alter and delete accounts from the database. They should assign and manage permissions to accounts to allow access to required resources. These resources should be the learner's own, as well as those that are to be shared with others.

Transaction processing

Teach learners the principles of database transactions and why they are important. They should group multiple statements into a transaction and make sure it is COMMITTED both manually or automatically by a SQL statement. You should also teach learners how transactions can be manually or automatically rolled back in the event of a failure and how SAVEPOINTS can facilitate partial rollbacks.

Use NoSQL databases

Learners should understand the differences between a traditional relational database and the common types of the more modern non-relational NoSQL databases.

They should be aware of how different types of NoSQL databases work and the methods they use to store, update and allow access to their data.

Learners should implement at least one type of NoSQL database. They should also extract information from it, as well as update, delete, import and export its data.

Approaches to assessment

Regardless of whether multiple assessments are used or if a holistic approach is taken to assessment, the assessment should cover the scope outlined in the evidence requirements.

Outcome 1

You could give learners an existing data set that they normalise into a set of tables in 3NF and then verify by producing an ER diagram from the data that they transform into an equivalent set of tables. Conversely, learners could create the ERD relational schema first and then verify as conforming to 3NF.

Outcomes 2, 3, 4, 5 and 6

Learners should provide evidence of the queries used to generate and populate the required set of tables from a database design. These should include database object creation and application of appropriate constraints.

Learners should provide evidence that they can create queries to extract analysed data to match those listed in the evidence requirements. This evidence should also include appropriate user-defined functions.

If you use a holistic approach, the database implementation could be the design learners have already created. Otherwise, you should provide an alternative design. Using the holistic approach, the created queries implement the design and allow learners to extract information from it. If you use a more atomic approach, learners should build up a portfolio of smaller examples for assessment.

Outcome 7

Learners should create a number of user accounts with different permissions on different database objects, such as tables and views. They should then test these accounts.

Outcome 8

Learners should create a transaction with invalid statements so that the transaction fails, and the database rolls back to a stable state.

Outcomes 9, 10 and 11

Learners should understand the principles behind the four common categories of NoSQL databases: key-value stores, wide column stores, document databases and graph databases. They should understand the advantages and limitations of NoSQL databases against relational databases.

Learners should then create a NoSQL database using at least one of the four categories and learn how to add, update and delete data, as well as query data from it. Learners could also extract some of the data from their relational database and upload it into a NoSQL database.

Equality and inclusion

This unit is designed to be as fair and as accessible as possible with no unnecessary barriers to learning or assessment.

You should take into account the needs of individual learners when planning learning experiences, selecting assessment methods or considering alternative evidence.

Guidance on assessment arrangements for disabled learners and/or those with additional support needs is available on the assessment arrangements web page: www.sqa.org.uk/assessmentarrangements.

Information for learners

Database Design and Development (SCQF level 8)

This information explains:

- what the unit is about
- what you should know or be able to do before you start
- what you need to do during the unit
- opportunities for further learning and employment

Unit information

This unit provides you with the knowledge and skills you need to design a relational database and then access and analyse its data. It also gives you important knowledge and skills in the development and data analysis of NoSQL databases. It is intended for any who have an interest in database design and data analysis. It is particularly suitable if you are studying topics in software development or data science.

The unit does not require any previous knowledge of database design or development. However, you should have a basic understanding of what a database is and why it is essential to modern business.

You are assessed by designing and implementing both a relational and non-relational database and then using a wide variety of techniques to extract data from it according to a specification.

When you finish the unit, you have gained important knowledge in database design and development. You can progress from the unit to more specialised vendor certifications in SQL.

The unit also provides ample opportunity for you to enhance your self-management, social intelligence, and innovative skills. You have several opportunities during the unit to facilitate this.

Administrative information

Published: June 2023 (version 1.0)

Superclass: CB

History of changes

Version	Description of change	Date

Please check SQA's website to ensure you are using the most up-to-date version of this document.