# Sql interview

An ETL workflow is responsible for the extraction of data from the source systems, their cleaning, transformation, and loading into the target data warehouse. There are existing formal methods to model the schema of source systems or databases such as entity-relationship diagram (ERD)

SQL Server Reporting Services (SSRS) is a server-based report generating software system from Microsoft. It is part of a suite of Microsoft SQL Server services, including SSAS (SQL Server Analysis Services) and SSIS (SQL Server Integration Services).

Administered via a Web interface, it can be used to prepare and deliver a variety of interactive and printed reports. The SSRS service provides an interface into Microsoft Visual Studio so that developers as well as SQL administrators can connect to SQL databases and use SSRS tools to format SQL reports in many complex ways. It also provides a 'Report Builder' tool for less technical users to format SQL reports of lesser complexity.

SSRS competes with Crystal Reports and other business intelligence tools.

Database refactoring

From Wikipedia, the free encyclopedia

Jump to navigationJump to search

A database refactoring is a simple change to a database schema that improves its design while retaining both its behavioral and informational semantics. Database refactoring does not change the way data is interpreted or used and does not fix bugs or add new functionality. Every refactoring to a database leaves the system in a working state, thus not causing maintenance lags, provided the meaningful data exists in the production environment.

A database refactoring is conceptually more difficult than a code refactoring; code refactorings only need to maintain behavioral semantics while database refactorings also must maintain informational semantics.

You refactor a database schema for one of several reasons:

To develop the schema in an evolutionary manner in parallel with the evolutionary design of the rest of your system.

To fix design problems with an existing legacy database schema. Database refactorings are often motivated by the desire for database normalization of an existing production database, typically to "clean up" the design of the database.

To implement what would be a large (and potentially risk) change as a series of small, low-risk changes.
Examples of database refactoring:
Splitting an aggregate table into two different tables in the process of .
Renaming an existing column to make its purpose clearer.
Combining two columns into a single one because they were being used for the same purpose.
Splitting an existing column into two or more columns because the original column was being used for several purposes (so you have one column per purpose).
Applying a common data format to a column so as to increase the consistency of the data.
Common code refactorings (Rename Method, Introduce Variable, Rename Variable, and so on) to database code such as stored procedures and triggers.
Introducing a view to implement a common access path to data.
https://en.wikipedia.org/wiki/Evolutionary_database_design
Evolutionary database design involves incremental improvements to the database schema so that it can be continuously updated with changes, reflecting the customer's requirements. People across the globe work on the same piece of software at the same time hence, there is a need for techniques that allow a smooth evolution of database as the design develops. Such methods utilize automated refactoring and continuous integration so that it supports agile methodologies for software development. These development techniques are applied on systems that are in pre-production stage as well on systems that have already been released. These techniques not only cover relevant changes in the database schema according to customer's changing needs, but also migration of modified data into the database and also customizing the database access code accordingly without changing the data semantics
Evolutionary database design aims to construct the database schema over the course of the project instead of building the entire database schema at the beginning of the project. This method of database design can capture and deal effectively with the changing requirements of projects.

There are five evolutionary database design techniques that can aid developers in building their database in an iterative fashion. A brief overview about the five techniques are provided below.

As mentioned in the previous section evolutionary methods are iterative in nature and these methods have become immensely popular over last two decades. Evolutionary database design aims to construct the database schema over the course of the project instead of building the entire database schema at the beginning of the project. This method of database design can capture and deal effectively with the changing requirements of projects.

There are five evolutionary database design techniques that can aid developers in building their database in an iterative fashion. A brief overview about the five techniques are provided below.

Database refactoring

Main article: Database refactoring

Refactoring is the process of making changes to the program without affecting the functionality of the program. Database refactoring is the technique of implementing small changes to the database schema without affecting the functionality and information stored in the database.[2] The main purpose of database refactoring is to improve the database design so that the database is more in-sync with the changing requirements. The user can modify tables, views, stored procedures and triggers. Dependency between the database and external applications make database refactoring a challenge.

Evolutionary data modeling

Data modeling is the technique of identifying entities, associating attributes to the entities and deciding the data structure to represent the attributes.[3] In the traditional database scenario, a logical data model is created at the beginning to represent the entities and their associated attributes. In evolutionary data modeling the technique of data modeling is performed in an iterative manner, that is multiple data models are developed, each model representing a different aspect of the database. This kind of data modeling technique is practiced in an agile environment and it is one of the main principles of agile development.[4]

Database regression testing

Whenever a new functionality is added to a system, it is essential to verify that the update does not corrupt or render the system unusable. In a database, the business logic is implemented in stored procedures, data validation rules and referential integrity and they have to be tested thoroughly when any change is implemented in the system. Regression testing is the process of executing all the test cases whenever a new feature is added to the system. test-first development (TFD) is a form of regression testing followed in evolutionary database design. The steps involved in TFD approach are,[3]

Before adding a new function to the system, add a test to the test case suite such that the system fails the test

Run the tests, either the entire set of test cases or just a subset and ensure that the newly added test does indeed fail

Update the function such that the test passes

Run the tests again to ensure that all they succeed and that the system is not broken

Configuration management of database artifacts

Configuration management is a detailed recording of versions and updates that have been applied to any system. Configuration management is useful in rolling back updates and changes which have impacted the system in a negative manner. To ensure that any updates made in database refactoring can be rolled back, it is important to maintain database artifacts like data definition language scripts, data model files, reference data, stored procedures, etc. in a configuration management system.[5]

Developer sandboxes

Main article: Sandbox (software development)

A sandbox is a fully functional environment in which the system can be built, tested and executed. In order to make changes to the database schema in an evolutionary manner it is ideal for every developer to have his/her own physical sandbox, copy of source code and a copy of database. In a sandbox environment the developer can make changes to the database schema and run tests without affecting the work of other developers and other environments. Once the change has been implemented successfully, it is promoted to pre-production

environment where in acceptance testing is performed and after the acceptance tests succeed it is deployed into production.

Advantages and disadvantages

Advantages

High quality of database design: In evolutionary database design, the developer makes small changes to the database schema in an incremental manner and this achieves a highly optimized database schema.

Handling change: In a traditional database approach, a lot of time is spent in remodeling and restructuring the database when the requirements change. In evolutionary database technique, the schema of the database is adjusted periodically to keep up with the changing requirements. Hence, evolutionary database design technique is better suited in handling the changing requirements.

Guaranteed working of system at all times: The evolutionary database design approach follows test-first development model, in which the complete working of a system is tested before and after implementing an update. Hence, it is guaranteed that the system always works.

Compatible with software development: The IT industry is progressing towards agile method of software development and evolutionary database design ensures that data development is in sync with software development.

Reduced overall effort: In an evolutionary environment only the functionality that is required at that moment is implemented and no more.

Disadvantages

Cultural impediments: Evolutionary database design approach is relatively a newer concept and many well qualified data professionals still advocate the traditional approach. Therefore, most of the databases are still being designed in a serial fashion and evolutionary database design is yet to gain support and traction among experienced data professionals.

Requires a learning curve: Most of the developers are more familiar with the traditional approach and it takes time to learn evolutionary design as it is not intuitive.

Complex: When the database has many external dependencies, making changes to the schema becomes all the more complicated as the external dependencies should also be updated to cope up with the changes made in the database

schema. With the increase in number of dependencies, Evolutionary Database Design approach becomes extremely complex.

Comparison with traditional database design

Traditional database design technique does not support changes like evolutionary database design technique.'Unfortunately, the traditional data community assumed that evolving database schema is a hard thing to do and as a result never thought through how to do it.'[1] In a way, the evolutionary design is better for application developers and traditional design is better for data professionals.[6]

| Properties | Traditional Database Design | Evolutionary Database Design |
|---|---|---|
| Design | Traditional databases were developed by collaboration between business analysts and users. | Evolutionary Databases were designed by software developers and data professionals. |
| Design Issues | They demonstrate some design issues in databases. Commercially available databases were developed by experienced individuals, but are now being serviced by the database and not data professionals.[6] | They are developed by a close alliance of software developers and data professionals. The database evolves with the development and hence is processed by the same individuals who were responsible for development. |
| Approach towards change | Any change requested by the user is incorporated in the logical model followed by the physical model and then tested to ensure perfect functionality.[6] | Change is an integral part of evolutionary database design. Any change requested by the user is immediately implemented in the database as well as the code. The data migration scripts need to be updated as well. |
| Dependency on ER diagram | Traditional design is methodological and due to its dependency on ER diagram and its detailed design phases such as user, logic, and physical we can track data as well as its meaning.[6] | Design is interleaved between stages for evolutionary database design. Thus, the relationship between entities can change over software development cycle and so does the ER diagram. An ETL workflow is responsible for the extraction of data from the source systems, their cleaning, transformation, and loading into the target data warehouse. There are existing formal methods to model the schema of source systems or databases such as entity-relationship diagram (ERD). Similarly, |

destination data warehouse can follow well-accepted standard data models such as star schema and snowflake schema. For databases, we have a well established relational algebra, but there is no equivalent algebra for ETL workflows. Contrary to source and target areas, models of ETL workflow are still in the fancy stage.