Click to verify

Share — copy and redistribute the material in any medium or format for any purpose, even commercially. Adapt — remix, transform, and build upon the material for any purpose, even commercially. The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit , provide a link to the license, and indicate if changes were made . You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. You do not have to comply with the license for elements of the material in the public domain or where your use is permitted by an applicable exception or limitation . No warranties are given. The license may not give you all of the permissions necessary for your intended use. For example, other rights such as publicity, privacy, or moral rights may limit how you use the material. Power BI makes it easy to visualize your data, but first, you need to connect it to SQL Server. This guide walks you through the process step -by -step. With SQL Spreads Excel Add-In, business users can bring in and update SQL Server data themselves — all from Excel — ensuring that your Power BI dashboards include all necessary data. Building dashboards and reports in Power BI requires two key steps: Get and transform your data — This involves getting all your data into a SQL Server database and specifically, how to connect Power BI to SQL Server. Design your dashboards and reports — Once your data is ready, you can create the visualizations your team needs. This guide focuses on the first step: getting your data into SQL Server and connecting it to Power BI. We'll walk you through the process, making sure you can centralize your data and use it effectively in Power BI. Here's a preview of the steps we'll cover in this guide: By the end of this guide, you'll have the knowledge to connect SQL Server to Power BI and keep your dashboards accurate and up to date. To demonstrate this in practice, let's consider a use case: creating a KPI dashboard in Power BI using data collated into a single SQL Server table. KPI dashboards often require data from multiple, scattered sources, and while many organizations aim to automate the process of gathering, calculating, and collating KPIs, much of it is still done manually — often in Excel. This is our sample data that we will use: This set of KPIs relates to customer service, with target values and monthly updates for actual values and comments on variances. Each month, the Customer Services Manager updates this data and presents the results to upper management through a dashboard. Centralizing the data in SQL Server makes it easier to manage, keep up to date, and integrate into your Power BI reports. We're going to create the data in an Excel spreadsheet and then import it into a table in SQL Server using the SQL Spreads Add-in for Excel. Here's how SQL Spreads makes this step easy: It's an Excel Add-In that connects to SQL Server. No coding required: set up is quick and easy, with out of the box functionality, saving you both time and money. An intuitive interface using familiar Excel. Once your table is created, you can share the document with your business users who can start to update the data in the SQL Server table as often as required. Worried about users messing up your data? With SQL Spreads, Designers control everything, while Data Editors are limited to safe, specific edits. Find out more about the Designer and Data Editor roles. First, install the SQL Spreads Add-in for Excel. You can download a free trial of SQL Spreads from here. Run the SQLSpreadsSetup.exe file and follow the instructions. Restart Excel and accept the Add-In confirmation. You will now find SQL Spreads in the tab menu in Excel: Next, we need to create our data in Excel. There are 8 KPIs that we're going to report on each month. Here is a copy of the Excel file if you want to follow along. Once the data has been created in Excel, we can use the SQL Spreads 'Table Creator' feature to create the table in SQL Server. The SQL Spreads' "Table Creator" feature is a much easier way to create tables in SQL Server. Convert the data that you've added into an Excel table by simply selecting one of the data cells and pressing CTRL + T. Click the Create new SQL Table button. If this is the first time you are using SQL Spreads or the Table Creator, you will see the dialog to connect to your SQL Server.Fill out your server details and click OK. Once you are connected to SQL Server, you will see the Create SQL Server Table dialog. Enter a table name and choose which SQL Server database to create the table in. SQL Spreads will always remember the last used table. If you need to change to another SQL Server click the Edit button next to the server name. Click OK. A confirmation message will appear confirming the table was created and how many rows of data were inserted into the table.We can check that the new table has been created and that our data is there by running a query in SQL Server Management Studio. You can now close Excel. In Step 3 later on, we'll explain how to use SQL Spreads to connect to the table in SQL Server that we just created and perform updates in Excel that we can save back to SQL Server. The next step is to connect Power BI to SQL Server and create a Power BI report. Open Power BI Desktop, click on 'Get Data' and then 'Import Data from SQL Server'. In the dialog, select the Server and Database (optional). The Data Connectivity mode defaults to 'Import' – leave it on this option. The import option actually loads the data into the Power BI file, making it quicker to work on, but is not suitable for very large datasets. The alternative Direct Query option doesn't load the data into Power BI, but instead runs queries against the source data each and every time a visualization is refreshed or a filter changed You can now select the SQL table that we just created. We won't do any transformations in this example, so you can click 'Load'. Once the data has been loaded, you'll see the tables in the Fields pane on the right-hand side. We also have a blank canvas to work with and the Visualizations pane to select our charts and other visualization types from. We can now go ahead and create the visualizations that we need for our report. Check out our earlier article on creating reports using Power BI for more information on how to add the most common visualization types. Here's a simple dashboard based on the sample customer service KPI data: We now need to publish the report to the web-based Power BI Service so that our business users can access the report. Make sure you have saved your report and then from the Home tab, click on Publish > My Workspace > Select. Sign in to the Power BI service if you're asked to do so. Once the report has been published you can click on the link in the dialog box. In Power BI Desktop, we connected directly to our on-premises SQL Server database. However, because the Power BI service is a cloud-based app, it needs a data gateway to act as a bridge between the cloud and our on-premises network. We therefore need to create a gateway on a local machine, add our SQL Server database as a data source, and then connect our Power BI dataset to this gateway. To install a gateway on your local machine, follow the instructions here – for testing purposes you can install the 'On-premises data gateway (personal mode)' option. Once you've completed the process of installing and configuring the gateway you can review details and troubleshoot issues by clicking on the ellipsis to the right of a dataset name and selecting 'Settings', or by viewing from the My Workspace > Datasets page. Now that we have our Power BI dataset connected to our on-premises SQL Server database via the gateway, we need to create a refresh schedule. The refresh schedule means that our report will remain up to date as the source data changes. Go to My Workspace > Datasets. Select the ellipsis (. . .) for your dataset, then select Schedule refresh. You can select the refresh frequency on a regular schedule, e.g. daily, or customize to several time slots. Normal Power BI subscription allows: up to 8 daily time slots Power BI Premium subscription allows: 48 time slots Enabling the scheduled refresh feature also allows you to perform a manual refresh on-demand (note that doing this will contribute to the count of daily refreshes you are allowed). To perform a manual refresh, select the dataset in the navigation pane, click the ellipses menu and select 'Refresh Now'. In the previous steps, we loaded our data into a SQL table and created a report in Power BI. We're now going to look at the process for ensuring that the data for our Power BI report is up to date. Open Excel and click on the Design Mode button in the SQL Spreads menu. A list of databases will be displayed, and you can select the Demos database and then the kpi_demo table. When you select the table, SQL Spreads will connect to SQL Server and import the data from the table into Excel. With the data loaded in Excel, we can now make changes to any of the existing records or add new rows. In our example, we're going to assume that we're submitting the report for the month of December. We therefore need to copy the rows for each of the KPIs from November and paste them as new rows at the bottom of the table. We can then change the month for each of the new rows to be December, and also add in the new actual KPI values and comments. Once you've made all the changes, click 'Save to Database' – all of the updates will be written to SQL Server. We can now go to our dataset in the Power BI service and perform a manual refresh. The KPIs for December, including the actual values and any comments that were added can now be viewed. In this article, we've shown how to connect Power BI to SQL Server. Often, a challenge associated with SQL Server data is that there is usually important data that lies outside of the database. It is easy to lose your mind trying to centralize this data yourself. With the SQL Spreads Excel Add-in, business users can easily update and insert this data into SQL Server directly from Excel, no complex development required. Once your data is in SQL Server, connecting and creating a Power BI report is quick and straightforward. The real value of SQL Spreads is how it lets your team update data in SQL Server from within Excel, with changes instantly reflected in your Power BI reports. This ensures all the necessary data is in SQL Server, streamlining your dashboard creation and reducing errors. To learn more about how SQL Spreads can help you with your data needs, download a free trial and get started straight away. Welcome to ONLC, your top destination for comprehensive IT training courses. We specialize in providing the highest quality education through both online training courses and IT group training sessions. Whether you're looking to advance your skills in AI, cyber security, obtain valuable certifications, or build your career from the ground up, ONLC has the perfect classes tailored to your needs. Our courses are designed by industry experts to ensure you receive up-to-date knowledge and hands-on experience. We make it easier than ever for you to achieve your professional goals, by empowering you with flexible training options so you can learn at your own pace and convenience. Join our community of successful learners and take the next step towards a thriving IT career with ONLC. Interactive Live Online Training: Learn from Home or ONLC Centers Experience unparalleled computer training with our live, instructor-led classes. Since 1983, we've been delivering top-notch, interactive training solutions designed for maximum learning efficiency. Join our live IT training sessions from any ONLC center across the nation for a well-equipped, interruption-free environment, or attend our online training courses from your home or office for added convenience. Regardless of your chosen location, our classes are fully interactive with a dedicated online instructor. You'll engage with the software, complete hands-on exercises, and build a strong technical foundation. Our ONLC instructors will monitor your progress, assist you on your computer for personalized assistance, and are readily available to answer questions and facilitate discussions. Moreover, our small class sizes guarantee individual attention and ample practice time. We also provide comprehensive team training programs. Businesses can explore tailored tech training solutions for their entire workforce. "...It was very effective and therefore I would definitely be interested in attending other classes in the same format. The instructor was very knowlegeable and provided a wealth of information about the current version, especially since the last version I used was several releases ago." "...I love all the classes I have taken through ONLC. All the instructors are kind and patient. They are very experienced in the programs they are teaching. I have recommended this site to all my friends, family, and employer. I look forward to taking more classes from ONLC." "One of the best training classes I have done. I really like the hands on instructions with the instructors screen beside mine and the phone/com was great to have on hand for questions." "I would strongly recommend ONLC courses to other individuals who require additional software skills with limited time availability. The classes are exemplary and the class size allows for individual attention which provides a high rate of comprehension." "I've been to 5-6 ONLC classes and each and every instructor has been amazing. I can't say enough good things about ONLC...for me it is the only way to learn. Thank you!" "I am so impressed with ONLC - the courses; the manuals which are actually usable, readable and understandable; and the instructors. If I ever need additional training, you will be my first choice!" Find Your Course ONLC On-Demand: Elevate your Career with Self-study Courses ONLC On-Demand courses empower you to learn new skills and technology at your own pace, enhancing your professional development. These courses are perfect for those who can't take time off from work, prefer self-paced learning, or need to focus on specific skills. ONLC On-Demand self-study courses are available for: Prepare for Computer Certifications with ONLC's Training Classes Earning a professional computer certification validates skills and knowledge. Achieving a professional computer certification validates your skills and knowledge. An industry-recognized technology certification can lead to higher salaries and provide advancement opportunities for working professionals. For those seeking employment, a computer certification can set you apart from other candidates in careers spanning business, management, and IT. The consensus is that the best way to prepare for a technical certification exam is through hands-on experience with the relevant technology and content. ONLC's IT training courses offer precisely that and more. Whether you choose our live, instructor-led courses or our self-study ONLC On-Demand online training courses, you'll engage in extensive labs and exercises* to reinforce your learning and understanding. Our training solutions equip you with the tools, skills, and resources necessary to enhance your learning experience. Many of our certification courses are vendor-authorized and globally recognized, including those from CompTIA, Microsoft, and Cisco. Elevate your business expertise and prove your proficiency by earning a professional computer certification today with training from ONLC. Learn more professional certifications here. Some of the most popular and in-demand computer certifications include: *Live, Instructor-led classes not addressing specific software like Business Process courses will have less of a hands-on component and will be more lecture/presentation delivery. On-Demand components can vary by title. ONLC is a business-to-business training provider. "ONLC is a trademark of Online Consulting, Inc. All other trademarks mentioned in this document or Web site are the property of their respective owners." ITIL® trademark notices iMicrosoft SQL Server - US (English). (2018). SQL Server 2017 on Windows and Linux | Microsoft. [online] Available at: [Accessed 8 May 2018]. iiSpenik, M; Sledge O (2000). Microsoft SQL Server 2000 DBA survival guide. SAMS. Chapter 2. ISBN 0672324687. iiiHarris, Scott; Curtis Preston (2007). Backup & Recovery: Inexpensive Backup Solutions for Open Systems. O'Reilly. p. 562. ISBN 0596102461. ivDocs.microsoft.com. (2018). Install SQL Server Database Engine. [online] Available at: [Accessed 8 May 2018]. vDocs.microsoft.com. (2018). SQL Server Integration Services. [online] Available at: [Accessed 9 May 2018]. viDocs.microsoft.com. (2018). What is SQL Server Reporting Services (SSRS). [online] Available at: [Accessed 9 May 2018]. viiWhigham, S. (2018). SQL Server 2008/R2 Analysis Services (Training Course) | LearnItFirst. [online] Learnitfirst.com. Available at: [Accessed 9 May 2018]. viiiMehta, S. (2018). SQL Server Analysis Services (SSAS) Tutorial. [online] Mssqltips.com. Available at: [Accessed 9 May 2018]. Skip to content Optimize your data integration with Hevo! Are you struggling with connecting Power BI to your SQL Server? Look no further, as this article will guide you through the process step by step. With the increasing demand for data analysis and visualization, knowing how to connect these two powerful platforms is essential for businesses to stay competitive. What Is Power BI? Power BI is a robust business intelligence tool developed by Microsoft that enables users to analyze and visualize data from multiple sources, including SQL Server databases. This powerful tool allows users to create interactive dashboards, reports, and data visualizations to gain valuable insights and make data-driven decisions. Its features include data modeling, transformation, and visualization capabilities. Additionally, Power BI seamlessly integrates with other Microsoft tools such as Excel and Azure, making it a comprehensive solution for data analysis and reporting. In summary, Power BI is a versatile and user-friendly tool for connecting, analyzing, and presenting data from SQL Server and other sources. What Is SQL Server? SQL Server, developed by Microsoft, is a relational database management system. Its main purpose is to store and retrieve data as requested by other software applications. SQL Server utilizes a structured query language (SQL) for managing and manipulating databases, providing a secure and scalable platform for data storage and access. It is widely used in various industries for data storage, analysis, and reporting. To connect Power BI to SQL Server, the Power BI Desktop tool can be used to establish a connection and import data for visualization and analysis. In summary, SQL Server is a powerful database management system that plays a crucial role in data-driven decision-making. It offers a robust and efficient platform for storing and managing large volumes of data. By following the necessary steps, you can easily connect Power BI to SQL Server and utilize the capabilities of both tools to gain valuable insights from your data. Why Connect Power BI to SQL Server? Why Connect Power BI to SQL Server? Connecting Power BI to SQL Server offers numerous benefits that enhance data analysis and reporting capabilities. These benefits include: Real-time insights Powerful visualizations and dashboards Advanced analytics Data governance By connecting Power BI to SQL Server, you can access real-time data directly from your database, enabling up-to-date insights and analysis. Additionally, Power BI integrates with SQL Server's advanced analytical capabilities, such as machine learning and predictive analytics, allowing you to uncover valuable insights. Moreover, connecting Power BI to SQL Server ensures data governance, as it centralizes and controls access to your data sources. In fact, connecting Power BI to SQL Server can significantly improve data-driven decision-making processes, leading to a 79% improvement in organizational performance. How to Connect Power BI to SQL Server? Are you looking to harness the power of Power BI to analyze your SQL Server data? Look no further! In this section, we will guide you through the process of connecting Power BI to SQL Server step by step. From installing Power BI Desktop to creating powerful visualizations, we'll cover all the necessary steps to get your data working for you. Let's dive in and learn how to connect Power BI to SQL Server for efficient data analysis. error Step 2: Connect to SQL Server To connect Power BI to SQL Server, follow these steps: Install Power BI Desktop. Launch Power BI Desktop and click on "Get Data". Choose the "SQL Server" data source and enter the server details. Select the desired database and click on "Connect". Choose the tables or views you want to import into Power BI. Apply any necessary data transformations and cleaning. Create visualizations using the imported data. Connecting Power BI to SQL Server is a crucial step in enabling real-time data analysis, interactive dashboards, easy data manipulation, and collaborative decision making. To optimize this connection, it is recommended to optimize the data source, use DirectQuery mode, utilize SQL Server views, schedule data refreshes, and ensure secure data access. Follow the below steps to connect to SQL Server: Step 3: Choose Data Source and Database When connecting Power BI to SQL Server, Step 3: Choose Data Source and Database is a crucial step in the process. This step involves selecting the specific data source and database that you want to connect to in Power BI. Launch Power BI Desktop and click on "Get Data" from the Home tab. In the Get Data window, choose "SQL Server" as the data source. Enter the server name or IP address in the Server box and select the desired authentication method. Click on the "Database" tab and choose the database you want to connect to. Click on "Connect" and Power BI will establish a connection to the selected SQL Server database. I recently used Power BI to connect to a SQL Server database for a client. After going through the process of choosing the data source and database, I was able to retrieve the relevant data and create visualizations that provided valuable insights for the client's business decisions. Step 4: Select Tables and Views After successfully connecting Power BI to SQL Server, the next step is to carefully select the relevant tables and views for data analysis and visualization. Open Power BI Desktop and navigate to the "Home" tab. Click on the "Get Data" button to access the various data sources. Choose the SQL Server option from the list of available connectors. In the "Navigator" window, select the desired database containing the necessary tables and views. Preview the data from each table or view to ensure its relevance. Select the tables and views required for the analysis by checking the corresponding checkboxes. Click on the "Transform Data" button to clean and manipulate the data, if needed. Proceed to create insightful visualizations using the selected tables and views. Fact: Power BI has the capability to connect to multiple tables and views simultaneously, allowing for comprehensive data analysis and valuable insights. Step 5: Transform and Clean Data Step 5: Transform and clean data is a crucial part of connecting Power BI to SQL Server: Perform data cleansing by removing duplicates, correcting errors, and handling missing values. Apply data transformations like filtering, sorting, and aggregating to prepare data for visualization. Use Power Query Editor to shape data by adding calculated columns, merging tables, or splitting columns. Apply data type conversions and formatting to ensure accurate and consistent data representation. Validate data transformations and clean data by previewing the results before loading into Power BI. Suggestions: Document data transformation steps for transparency and reproducibility. Implement automated data cleaning processes to maintain data integrity. Regularly review and update data transformation rules to adapt to changing business needs. Step 6: Create Visualizations To create visualizations in Power BI, follow these steps: Open Power BI Desktop and connect to the SQL Server database. Select the desired data source and database. Choose the tables and views that contain the data you want to visualize. Transform and clean the data as needed to prepare it for visualization. Use Power BI's wide range of visualization options to create interactive and informative visualizations. Add filters, slicers, and drill-through functionality to enhance the interactivity of your visualizations. Visualizations have been an integral part of data analysis for centuries. From the earliest bar charts and line graphs in the 18th century to today's advanced interactive visualizations, businesses and individuals have relied on visual representations to gain insights and make informed decisions based on data. With the advent of tools like Power BI, creating stunning visualizations and harnessing the power of data visualization has become accessible to everyone. What Are the Benefits of Connecting Power BI to SQL Server? In today's data-driven world, businesses are constantly looking for ways to analyze and utilize their data more efficiently. Connecting Power BI to SQL Server is a powerful tool that can help achieve this goal. In this section, we will discuss the various benefits of connecting Power BI to SQL Server, including real-time data analysis, interactive dashboards, easy data manipulation, and collaborative decision making. By the end, you'll have a better understanding of how this connection can enhance your data analysis capabilities. 1. Real-time Data Analysis Real-time data analysis is a crucial capability when connecting Power BI to SQL Server. To perform real-time data analysis in Power BI, follow these steps: Install Power BI Desktop. Connect to SQL Server. Choose the data source and database. Select tables and views. Transform and clean data. Create visualizations. By connecting Power BI to SQL Server, you can achieve several benefits, such as: Real-time data analysis. Interactive dashboards. Easy data manipulation. Collaborative decision making. For example, a manufacturing company utilized the capability of real-time data analysis by connecting Power BI to SQL Server to monitor their production data. With dynamic dashboards and interactive visualizations, they were able to identify and address bottlenecks, ultimately resulting in increased efficiency and cost savings. 2. Interactive Dashboards Interactive dashboards in Power BI provide users with dynamic and customizable data visualizations that allow for real-time insights and analysis. To create interactive dashboards in Power BI, follow these steps: Connect Power BI Desktop to SQL Server. Select the desired data source and database. Choose the relevant tables and views for your dashboard. Transform and clean the data as needed. Create visualizations using Power BI's intuitive drag-and-drop interface. By connecting Power BI to SQL Server, users can leverage the following benefits: Real-time data analysis for up-to-the-minute insights. Interactive dashboards that enable exploration and drill-down capabilities. Easy data manipulation through filtering, sorting, and slicing. Collaborative decision making by sharing dashboards and reports with stakeholders. To make the most of the interactive dashboards in Power BI, consider these best practices: Optimize the data source for efficient querying. Utilize DirectQuery mode for real-time data access. Incorporate SQL Server views to simplify data modeling. Schedule regular data refreshes to ensure accuracy. Secure data access by implementing appropriate user permissions and encryption. 3. Easy Data Manipulation Connecting Power BI to SQL Server allows for easy data manipulation and analysis. Here are the steps to connect Power BI to SQL Server: Install Power BI Desktop Connect to SQL Server Choose Data Source and Database Select Tables and Views Transform and Clean Data Create Visualizations Pro-tip: Make data manipulation even easier by utilizing the Power Query Editor in Power BI, which offers advanced features such as merging tables, filtering data, and creating calculated columns. 4. Collaborative Decision Making Collaborative decision making is a key benefit of connecting Power BI to SQL Server. Here are the steps to foster collaboration: Share dashboards: Publish your Power BI reports to the Power BI service and share them with your team members. Collaborate in real time: Use the Power BI service to collaborate with others by adding comments and annotations to reports. Enable data-driven discussions: Use the Q&A feature in Power BI to have interactive discussions around the data. Create and share insights: Use Power BI's sharing capabilities to distribute reports and insights with stakeholders. True story: A sales team used Power BI connected to SQL Server to analyze sales data collaboratively. By sharing dashboards and insights, the team was able to make data-driven decisions, identify opportunities, and increase sales by 15% within six months. What Are the Best Practices for Connecting Power BI to SQL Server? Connecting Power BI to SQL Server is crucial for creating insightful and dynamic visualizations. However, to ensure a smooth and efficient connection, it is important to follow best practices. In this section, we will discuss the top five best practices for connecting Power BI to SQL Server. From optimizing the data source to securing data access, we will cover all the necessary steps to establish a strong and reliable connection. Let's dive in and learn the most effective ways to connect these powerful tools. 1. Optimize Data Source To ensure efficient and effective data analysis when connecting Power BI to SQL Server, follow these steps to optimize your data source: Use indexed columns: By indexing columns in your database, you can improve query performance and speed up data retrieval. Leverage stored procedures: Instead of directly querying tables, utilize stored procedures to optimize data retrieval and enhance security. Create views: Views can simplify complex queries, improve data organization, and enhance overall performance. Filter data at the source: Apply filters to your SQL queries to reduce the amount of data transferred to Power BI, improving query performance. Aggregate data: Pre-aggregating data at the SQL Server level can optimize query performance by reducing the amount of data processed and transferred. By following these best practices, you can effectively optimize your data source when connecting Power BI to SQL Server. 2. Use DirectQuery Mode When connecting Power BI to SQL Server, utilizing DirectQuery mode offers numerous advantages. To implement this mode, simply follow these steps: Install Power BI Desktop. Connect to SQL Server. Choose the desired data source and database. Select the appropriate tables and views to include. Transform and clean the data as necessary. Create visualizations using the data. Using DirectQuery mode allows for real-time data analysis by retrieving data directly from the SQL Server database. It also enables interactive dashboards, easy data manipulation, and collaborative decision-making. By following best practices, such as optimizing the data source, utilizing SQL Server views, and scheduling data refreshes, the connection between Power BI and SQL Server can be maximized. 3. Utilize SQL Server Views To utilize SQL Server views in Power BI, follow these steps: Create a SQL Server view that contains the desired data and logic. In Power BI, click on "Get Data" and select the appropriate SQL Server connection. Choose the database that contains the view. Select the view from the list of available tables and views. Apply any necessary transformations and cleaning to the data. Create visualizations and reports using the view as the data source. By utilizing SQL Server views in Power BI, you can easily incorporate the desired data and logic into your reports. This simplifies data analysis, enhances performance, and ensures data consistency across different reports. Views also provide a layer of abstraction, allowing for easy modification of underlying data structures without impacting reports. Furthermore, views can improve security by limiting access to specific columns or rows of data. 4. Schedule Data Refresh To schedule data refresh in Power BI, follow these steps: Open Power BI Desktop and go to the "Home" tab. Click on "Transform data" to open the Power Query Editor. In the Power Query Editor, make any necessary transformations and cleaning steps. Once the data is prepared, click on "Close & Apply" to load it into Power BI. Go to the "File" menu and select "Options and settings" and then "Data Source Settings". In the "Data Source Settings" window, select the data source that requires scheduled refresh. Click on "Schedule…" and set the desired refresh frequency and time. Click on "OK" to save the changes and close the window. Finally, publish your report to the Power BI service to enable the scheduled data refresh. 5. Secure Data Access To ensure secure data access when connecting Power BI to SQL Server, follow these steps: Implement strong authentication protocols, such as using Windows authentication or Azure Active Directory. Encrypt data transmission between Power BI and SQL Server using SSL/TLS protocols. Restrict user access to SQL Server by assigning appropriate permissions based on roles and responsibilities. Regularly update and patch SQL Server to address any security vulnerabilities. Monitor and audit SQL Server activity to detect and respond to any suspicious or unauthorized access attempts. Digital database whose organization is based on the relational model of data This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.Find sources: "Relational database" – news · newspapers · books · scholar · JSTOR (January 2024) (Learn how and when to remove this message) A relational database (RDB[1]) is a database based on the relational model of data, as proposed by E. F. Codd in 1970.[2] A Relational Database Management System (RDBMS) is a type of database management system that stores data in a structured format using rows and columns. Many relational database systems are equipped with the option of using SQL (Structured Query Language) for querying and updating the database.[3] The concept of relational database was defined by E. F. Codd at IBM in 1970. Codd introduced the term relational in his research paper "A Relational Model of Data for Large Shared Data Banks".[2] In this paper and later papers, he defined what he meant by relation. One well-known definition of what constitutes a relational database system is composed of Codd's 12 rules. However, no commercial implementations of the relational model conform to all of Codd's rules,[4] so the term has gradually come to describe a broader class of database systems, which at a minimum: Present the data to the user as relations (a presentation in tabular form, i.e. as a collection of tables with each table consisting of a set of rows and columns); Provide relational operators to manipulate the data in tabular form. In 1974, IBM began developing System R, a research project to develop a prototype RDBMS.[5][6] The first system sold as an RDBMS was Multics Relational Data Store (June 1976).[7][8][citation needed] Oracle was released in 1979 by Relational Software, now Oracle Corporation.[9] Ingres and IBM BS12 followed. Other examples of an RDBMS include IBM Db2, SAP Sybase ASE, and Informix. In 1984, the first RDBMS for Macintosh began being developed, code-named Silver Surfer, and was released in 1987 as 4th Dimension and known today as 4D.[10] The first systems that were relatively faithful implementations of the relational model were from: University of Michigan – Micro DBMS (1969)[11] Massachusetts Institute of Technology (1971)[12] IBM UK Scientific Centre at Peterlee – IS1 (1970–72),[13] and its successor, PRTV (1973–79).[14] The most common definition of an RDBMS is a product that presents a view of data as a collection of rows and columns, even if it is not based strictly upon relational theory. By this definition, RDBMS products typically implement some but not all of Codd's 12 rules. A second school of thought argues that if a database does not implement all of Codd's rules (or the current understanding on the relational model, as expressed by Christopher J. Date, Hugh Darwen and others), it is not relational. This view, shared by many theorists and other strict adherents to Codd's principles, would disqualify most DBMSs as not relational. For clarification, they often refer to some RDBMSs as truly-relational database management systems (TRDBMS), naming others pseudo-relational database management systems (PRDBMS).[citation needed] As of 2009, most commercial relational DBMSs employ SQL as their query language.[15] Alternative query languages have been proposed and implemented, notably the pre-1996 implementation of Ingres QUEL. Main article: Relational model A relational model organizes data into one or more tables (or "relations") of columns and rows, with a unique key identifying each row. Rows are also called records or tuples.[16] Columns are also called attributes. Generally, each table/relation represents one "entity type" (such as customer or product). The rows represent instances of that type of entity (such as "Lee" or "chair") and the columns represent values attributed to that instance (such as address or price). For example, each row of a class table corresponds to a class, and a class corresponds to multiple students, so the relationship between the class table and the student table is "one to many"[17] Each row in a table has its own unique key. Rows in a table can be linked to rows in other tables by adding a column for the unique key of the linked row (such columns are known as foreign keys). Codd showed that data relationships of arbitrary complexity can be represented by a simple set of concepts.[2] Part of this processing involves consistently being able to select or modify one and only one row in a table. Therefore, most physical implementations have a unique primary key (PK) for each row in a table. When a new row is written to the table, a new unique value for the primary key is generated; this is the key that the system uses primarily for accessing the table. System performance is optimized for PKs. Other, more natural keys may also be identified and defined as alternate keys (AK). Often several columns are needed to form an AK (this is one reason why a single integer column is usually made the PK). Both PKs and AKs have the ability to uniquely identify a row within a table. Additional technology may be applied to ensure a unique ID across the world, a globally unique identifier, when there are broader system requirements. The primary keys within a database are used to define the relationships among the tables. When a PK migrates to another table, it becomes a foreign key (FK) in the other table. When each cell can contain only one value and the PK migrates into a regular entity table, this design pattern can represent either a one-to-one or one-to-many relationship. Most relational database designs resolve many-to-many relationships by creating an additional table that contains the PKs from both of the other entity tables – the relationship becomes an entity; the resolution table is then named appropriately and the two FKs are combined to form a PK. The migration of PKs to other tables is the second major reason why system-assigned integers are used normally as PKs; there is usually neither efficiency nor clarity in migrating a bunch of other types of columns. Relationships are a logical connection between different tables (entities), established on the basis of interaction among these tables. These relationships can be modelled as an entity-relationship model. In order for a database management system (DBMS) to operate efficiently and accurately, it must use ACID transactions.[18][19][20] Part of the programming within a RDBMS is accomplished using stored procedures (SPs). Often procedures can be used to greatly reduce the amount of information transferred within and outside of a system. For increased security, the system design may grant access to only the stored procedures and not directly to the tables. Fundamental stored procedures contain the logic needed to insert new and update existing data. More complex procedures may be written to implement additional rules and logic related to processing or selecting the data. Relational database terminology The relational database was first defined in June 1970 by Edgar Codd, of IBM's San Jose Research Laboratory.[2] Codd's view of what qualifies as an RDBMS is summarized in Codd's 12 rules. A relational database has become the predominant type of database. Other models besides the relational model include the hierarchical database model and the network model. The table below summarizes some of the most important relational database terms and the corresponding SQL term: SQL term Relational database term Description Row Tuple or record A data set representing a single item Column Attribute or field A labeled element of a tuple, e.g. "Address" or "Date of birth" Table Relation or Base relvar A set of tuples sharing the same attributes; a set of columns and rows View or result set Derived relvar Any set of tuples; a data report from the RDBMS in response to a query Main articles: Relation (database) and Table (database) In a relational database, a relation is a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints. The relational model specifies that the tuples of a relation have no specific order and that the tuples, in turn, impose no order on the attributes. Applications access data by specifying queries, which use operations such as select to identify tuples, project to identify attributes, and join to combine relations. Relations can be modified using the insert, delete, and update operators. New tuples can supply explicit values or be derived from a query. Similarly, queries identify tuples for updating or deleting. Tuples by definition are unique. If the tuple contains a candidate or primary key then obviously it is unique; however, a primary key need not be defined for a row or record to be a tuple. The definition of a tuple requires that it be unique, but does not require a primary key to be defined. Because a tuple is unique, its attributes by definition constitute a superkey. Main articles: Relvar and View (database) All data are stored and accessed via relations. Relations that store data are called "base relations", and in implementations are called "tables". Other relations do not store data, but are computed by applying relational operations to other relations. These relations are sometimes called "derived relations". In implementations these are called "views" or "queries". Derived relations are convenient in that they act as a single relation, even though they may grab information from several relations. Also, derived relations can be used as an abstraction layer. Main article: Data domain A domain describes the set of possible values for a given attribute, and can be considered a constraint on the value of the attribute. Mathematically, attaching a domain to an attribute means that any value for the attribute must be an element of the specified set. The character string "ABC", for instance, is not in the integer domain, but the integer value 123 is. Another example of domain describes the possible values for the field "CoinFace" as ("Heads","Tails"). So, the field "CoinFace" will not accept input values like (0,1) or (H,T). Constraints are often used to make it possible to further restrict the domain of an attribute. For instance, a constraint can restrict a given integer attribute to values between 1 and 10. Constraints provide one method of implementing business rules in the database and support subsequent data use within the application layer. SQL implements constraint functionality in the form of check constraints. Constraints restrict the data that can be stored in relations. These are usually defined using expressions that result in a Boolean value, indicating whether or not the data satisfies the constraint. Constraints can apply to single attributes, to a tuple (restricting combinations of attributes) or to an entire relation. Since every attribute has an associated domain, there are constraints (domain constraints). The two principal rules for the relational model are known as entity integrity and referential integrity. Main articles: Unique key and Primary key Every relation/table has a primary key, this being a consequence of a relation being a set.[21] A primary key uniquely specifies a tuple within a table. While natural attributes (attributes used to describe the data being entered) are sometimes good primary keys, surrogate keys are often used instead. A surrogate key is an artificial attribute assigned to an object which uniquely identifies it (for instance, in a table of information about students at a school they might all be assigned a student ID in order to differentiate them). The surrogate key has no intrinsic (inherent) meaning, but rather is useful through its ability to uniquely identify a tuple. Another common occurrence, especially in regard to N:M cardinality is the composite key. A composite key is a key made up of two or more attributes within a table that (together) uniquely identify a record.[22] Main article: Foreign key Foreign key refers to a field in a relational table that matches the primary key column of another table. It relates the two keys. Foreign keys need not have unique values in the referencing relation. A foreign key can be used to cross-reference tables, and it effectively uses the values of attributes in the referenced relation to restrict the domain of one or more attributes in the referencing relation. The concept is described formally as: "For all tuples in the referencing relation projected over the referencing attributes, there must exist a tuple in the referenced relation projected over those same attributes such that the values in each of the referencing attributes match the corresponding values in the referenced attributes." Main article: Stored procedure A stored procedure is executable code that is associated with, and generally stored in, the database. Stored procedures usually collect and customize common operations, like inserting a tuple into a relation, gathering statistical information about usage patterns, or encapsulating complex business logic and calculations. Frequently they are used as an application programming interface (API) for security or simplicity. Implementations of stored procedures on SQL RDBMS's often allow developers to take advantage of procedural extensions (often vendor-specific) to the standard declarative SQL syntax. Stored procedures are not part of the relational database model, but all commercial implementations include them. Main article: Index (database) An index is one way of providing quicker access to data. Indices can be created on any combination of attributes on a relation. Queries that filter using those attributes can find matching tuples directly using the index (similar to Hash table lookup), without having to check each tuple in turn. This is analogous to using the index of a book to go directly to the page on which the information you are looking for is found, so that you do not have to read the entire book to find what you are looking for. Relational databases typically supply multiple indexing techniques, each of which is optimal for some combination of data distribution, relation size, and typical access pattern. Indices are usually implemented via B+ trees, R-trees, and bitmaps. Indices are usually not considered part of the database, as they are considered an implementation detail, though indices are usually maintained by the same group that maintains the other parts of the database. The use of efficient indexes on both primary and foreign keys can dramatically improve query performance. This is because B-tree indexes result in query times proportional to log(n) where n is the number of rows in a table and hash indexes result in constant time queries (no size dependency as long as the relevant part of the index fits into memory). Main article: Relational algebra Queries made against the relational database, and the derived relvars in the database are expressed in a relational calculus or a relational algebra. In his original relational algebra, Codd introduced eight relational operators in two groups of four operators each. The first four operators were based on the traditional mathematical set operations: The union operator (∪) combines the tuples of two relations and removes all duplicate tuples from the result. The relational union operator is equivalent to the SQL UNION operator. The intersection operator (∩) produces the set of tuples that two relations share in common. Intersection is implemented in SQL in the form of the INTERSECT operator. The set difference operator (-) acts on two relations and produces the set of tuples from the first relation that do not exist in the second relation. Difference is implemented in SQL in the form of the EXCEPT or MINUS operator. The cartesian product (X) of two relations is a join that is not restricted by any criteria, resulting in every tuple of the first relation being matched with every tuple of the second relation. The cartesian product is implemented in SQL as the Cross join operator. The remaining operators proposed by Codd involve special operations specific to relational databases: The selection, or restriction, operation (σ) retrieves tuples from a relation, limiting the results to only those that meet a specific criterion, i.e. a subset in terms of set theory. The SQL equivalent of selection is the SELECT query statement with a WHERE clause. The projection operation (π) extracts only the specified attributes from a tuple or set of tuples. The join operation defined for relational databases is often referred to as a natural join (⋈). In this type of join, two relations are connected by their common attributes. MySQL's approximation of a natural join is the Inner join operator. In SQL, an INNER JOIN prevents a cartesian product from occurring when there are two tables in a query. For each table added to a SQL Query, one additional INNER JOIN is added to prevent a cartesian product. Thus, for N tables in an SQL query, there must be N−1 INNER JOINS to prevent a cartesian product. The relational division (÷) operation is a slightly more complex operation and essentially involves using the tuples of one relation (the dividend) to partition a second relation (the divisor). The relational division operator is effectively the opposite of the cartesian product operator (hence the name). Other operators have been introduced or proposed since Codd's introduction of the original eight including relational comparison operators and extensions that offer support for nesting and hierarchical data,