ComputerWeekly.com

Guide to sourcing Database Developers

How to source the right database developers for your business

CONTENTS

ABOUT COMPUTER WEEKLY

ComputerWeekly.com is the number one online destination for senior IT decision-making professionals.

It is dedicated to providing IT professionals with the best information, the best knowledge and the best range of solutions that will enable them to succeed in the industry.

- ComputerWeekly.com benefits from Computer Weekly magazines unrivalled 40 year history

- ComputerWeekly.com offers exposure to a senior IT audience, backed by user profile research

- Computer Weekly is a five-times winner of the PPA Editorial Campaign of the Year award, demonstrating editorial excellence

- ComputerWeekly.com produces editorially independent breaking news picked up regularly by the media

- Initiatives such as the CW500 club reinforces Computer Weekly's impact and influence amongst senior IT decision-makers

- Complete dominance of the national news agenda – over 750 mentions within the media in 2006

ABOUT APPROVED INDEX

Computer Weekly Guides are provided courtesy of **Approved Index Ltd.**, the UK's number one online B2B buyer's tool. Approved Index offers free, quotes and buying advice across a range of over 100 product and service categories. Whether you're looking for office equipment, marketing services or corporate training, visit www.ApprovedIndex.co.uk to make the smartest purchasing decisions for your business.

If you have any questions regarding our service, just contact the Approved Index team on **0800 6122 113.**

DATABASE DEVELOPERS: A GUIDE

Living in the so-called "Information Age" dictates that information has a high value and that most businesses cannot function without a means of storing and managing data. Thus, industry and society in general demand not only the structuring of information, but highly effective and specialised systems of interaction with it. Databases provide the solution, and basically refer to places of stored information.

This guide will give you an introduction to the world of databases and help you define your requirements and find the best database developers for your company.

INTRODUCTION TO DATABASES

A database is a system which stores information using computer systems in a useful, structured way – a bit like an electronic filing system or library. Databases provide a means of storing, categorising, searching and analysing large amounts of similar data and are central to many business processes today. Whenever you telephone a company and they look up your account details, whenever you use a credit card or a cash machine, behind the scenes a database is in use.

Wherever a large amount of similar data (for example, different customers' personal details) is stored, some kind of database is the most appropriate system. Unless the number of records (for example, a single customer's details) is very small, it is the only sensible solution, as storing a separate file on a computer for each record outside of a database system is little better than using paper-based records and filing systems.

Databases are used on websites to store users' account information, news articles, blog posts, and product details in online shops. A database of postcodes and full postal addresses is used by Royal Mail to help deliver mail and allow businesses to look up addresses with just the post code. Governments use databases to gather census information and manage taxation and welfare systems. Companies uses databases to run loyalty card schemes, which in turn provides them with detailed sales information which can be used for strategy and marketing.

Whatever your business, you probably use some kind of database system already, and if not you will definitely benefit from one.

DATABASES DEMYSTIFIED

This guide is not intended to be a technical document, and a full technical understanding of databases is unnecessary for someone seeking database development services. It is useful to understand some of the basic concepts and terminology, however, to decrease the gap between database developers' knowledge and your own. Below is an explanation of some key concepts and a simple example database. A more detailed glossary can be found at the end of this guide.

Data and information

While in everyday language the two terms can be use synonymously, data and information mean subtly different things in the world of databases. Data means the raw facts, which could be numbers or pieces of text. On their own, individual pieces of data are meaningless. Information is data in some structured context – if we know what a particular number represents, for example the amount of money in a bank account, we have information which is the account's balance. Information can be processed and analysed.

Schemas

Put simply, a schema is a definition of the structure of the database. It determines how different kinds of information relate to one another and is the core of the database's design. For example, the schema could state that in one part of the database, records store customers' personal details (name, address etc.) while another part of the database stores each order made by a customer. Typically, each customer could make more than one order, but each order is **made by** only one customer.

Relational databases

Types of databases are defined by the kind of structure they have, and by far the most common type of database in use today is the relational database. These kinds of database are based on the mathematical relational model. In practice, most relational database systems use a common set of simplified, more obvious terminology for what are in fact mathematical concepts. Below is an explanation of a simple example database using this terminology.

A simple example

This simple example is based around a database for a shop. It could be an online shop or a physical shop and applies equally well to each, since we are not concerned with the details of the business, but only the data and how it is structured. Our database will store three distinct categories of information: customers, products, and orders.

Tables, rows and columns

The easiest way to think of a table in database terms is as a spreadsheet. A spreadsheet might list all of our customers and the table in our database for customers will be very similar. Each column would represent an attribute associated with this customer: First name, surname, address, telephone number, and so on. Each row of the table represents one customer. While the combination of all attributes for a customer is likely to be unique, it is common to add a column with an ID code, a unique number for each row or customer. This unique ID is known as a Primary Key and is the means by which we associate records or rows of one type with those of another.

Our table for products would similarly list various attributes for each product: A name, description, price, and an ID as a primary key to uniquely identify each product. Note that, for simplicity, each row in the fictitious products table represents one product line, as opposed to each individual item in stock. Although dealing with stock levels is a common feature of databases, it is also an extra layer of complexity and is ignored for the purposes of this example.

Last is our orders table. The purpose of this table is to record the fact of a customer placing an order for a product. As with our other tables, we would have a unique ID column, and to store the required information we also have a column for a customer's ID and a product's ID. In the context of the orders table, the customer and products IDs are referred to as Foreign Keys, since they are Primary Keys in a different (hence foreign) table. Each row in the orders tables relates a customer to a product, which constitutes an order. In a real-world system, your database would probably be able to handle orders for multiple products, but in this example an order is restricted to one product.

Relationships

As is implied by the name and the above descriptions, a fundamental element of relational databases is relationships. In our example, a customer may place one or more orders, each order is placed by only one customer and is for only one products. This is implicit since each row (i.e. each order) the orders table stores only one customer and one product. The relationship between customers and orders, and the relationship between products and orders, is called one-to-many, since one customer may place multiple orders, and one product may be part of many different orders.

Relationships can also be one-to-one. For example, a table of extended information about each customer like date of birth, could have one row for each customer, and be related to the main customers table on a one-to-one basis. Clearly this is similar to simply having more attributes in the main customers table, though splitting the data in this way may improve the performance of the database since information is only retrieved when necessary.

The third kind relationship is many-to-many. Imagine our database also stored information on the company's sales representatives and we wanted to record which representatives had dealt with which customers. This would be a many-to-many relationship, since a customer may have spoken to many representatives, and each representative would have spoken to many customers.

Queries

Now that our simple database's structure is defined, let's see how we can access the information stored in it. Clearly we could look up a customer or product based on their names or other attributes, and all information has an ID associated with it, but in order to make the most of our data we need queries.

A query is a sort of instruction to the database to retrieve information according to a give set of constraints. For example, if we wanted to find out all the products a particular customer had ordered, our query would find all of the rows in the products table, for which there is a corresponding row in the orders table, where the customer ID in the orders table is the ID of the customer we're interested in. Similarly we could list all the customers who have ordered a particular product or all the products which have never been ordered. More complicated queries can be used to count the number of orders per product, or orders per customer.

SQL

Queries are formulated in a programming language called a query language, and by far the most common of these is Structured Query Language (SQL, commonly pronounced 'sequel'). In SQL, a query to list the names of products order by a customer named John Smith might look like this:

```
SELECT products.name
FROM products
WHERE products.ID IN (
      SELECT orders.productID
      FROM orders
      WHERE orders.customerID IN (
         SELECT customers.ID
         FROM customers
         WHERE customers.Firstname = ’John’ AND
         customers.Surname=’Smith’)
      );
```

As you can see, designing and programming even a simple database can be very complicated. Luckily you don’t need to worry about this as a database system developed for you could have the functionality to find all of a customer’s orders built into it. However, it is useful to have an idea of what is going on behind the scenes. This is also a demonstration of how the structure of a database is of fundamental importance, since the right design means that queries like the above can be made (relatively) easily, quickly and efficiently. A badly designed and implemented database will be slow and make some queries impractical or even impossible.

Reports

A database will enable the retrieval of information of one or a small number of records, for example one customer’s details and order history, but an equally important function is reporting. In general, reports are a way of aggregating or summarising the information in a database. This could be as simple as a listing of all customers, but can also be used for listing outstanding orders, the top selling products, or stock levels. Reports are based on queries and can be very complicated but are way to analyse the information in your database. As with queries as degree of programming is usually involved.

ESTABLISHING YOUR REQUIREMENTS

Working out what you want from your database is the first step to getting the system you need. You probably have a good idea about what information you will be storing, but just as important is how you will access, manipulate and analyse it. Think about where the data will come from in the first place – will it be inputted manually (known as data entry)? Who will do this and how should it work? Try to list the different instances the database will be used in and have an idea of what should happen in each instance.

Database companies should be able to help formalise your needs into a detailed specification, but firmly establishing your own requirements means that suppliers can assess the task at hand and provide accurate quotations. There are a number of things to consider when considering your requirements.

Type of project

Building a new database system is not the only kind of database development project. Many companies have existing legacy systems which need to be updated or adapted. You may wish to integrate an existing database with a different application, for example a website, or have a new database system built to integrate with an existing application. You are probably clear about the type of your project, but you should be aware that there can be varying cost implications. Sometimes integrating an esoteric legacy system can be more costly than rebuilding it from scratch.

Stakeholders

Think about your organisation and who will use the database. You will probably find that different departments have very different needs and priorities for the system. It is probably worth including a representative from each department when discussing database requirements. Additionally, customer-facing staff will have different needs to middle and senior management teams. Those in customer service roles will be more concerned with accessing and editing individual customers' or client's information, whereas managers will probably need detailing reporting facilities.

Users and access levels

Considering who will use the database and how will naturally lead to matters of users and access levels. Assuming different people will use the database, there may be a case for having user accounts with different levels of access to data. For example, you may want staff in your accounts department to be the only people who can amend financial information. You may even want to make part or all of your database available to the general public, perhaps via a web interface. Make sure you have an idea of any requirements like this so that they can be included in the specification and implementation of your database from the outset.

Comprehensive requirements

Having discovered the different ways your database will be used you should now be able to exhaustively list all functionality to be included in the system. Think about what information you might want to print from the system and how it should look, how users will search the database, and how reports are to be generated. At this stage you can indulge in some creativity and list some features that, while not essential, could be useful. You may find later that implementing them is relatively easy for a database developer, and if not you won't be losing core functionality from your system.

Prioritise

Now that you have a lengthy set of potential functionalities and features, you should prioritise the list and decide which of these are essential, which are important and which would be nice to have, but not mission-critical. Doing this helps clarify your requirements, provides a sensible order for implementation and means that, should your time or budget be constrained, less important requirements can be dropped or left to be added later.

Custom-built and off the shelf databases

For many of the common uses of databases there are proven, ready-made database systems available. For example, Customer Relationship Management (CRM), inventories, accounting and online shops with product databases are all areas with established, off the shelf solutions available. If your needs could be met by one of these systems with little or no customisation, they will probably work out to be cheaper in the long run, even if the initial purchase price is high. On the other hand, if your business or your needs are very specific, the level of customisation required of an off the shelf system might be as much (in terms of time and cost) as a bespoke solution.

Technologies

As with most types of computer software, there is a range of competing and complementary systems available. Which is the right solution for your needs depends largely on the amount of data that will be stored. For example, Microsoft Access (which is included in Microsoft's Office suite) is perfect for a database with low numbers of records which will only be used by one or a handful of users, and allows sophisticated database applications to be created quickly. Unfortunately Access does not perform so well for larger databases, with multiple users or over networks. It can, however, be used to rapidly build a prototype for a larger system, or as a graphical front end which connects to a different back end database system.

Larger open source (i.e. free) databases include MySQL and PostgresSQL. Larger off the shelf databases that require licenses to use include Microsoft SQL Server, Oracle, or IBM DB2.

PRICE GUIDE

As with any computer project, the price you pay can vary hugely depending on the nature of your project and your exact requirements. Below is a rough guide, which should be viewed noting that suppliers may have a minimum threshold price for undertaking database work, as well as the fact that it is impossible to give precise costs without discussing your requirements with database developers.

Adapting an existing database

If a database developer is maintaining or modifying a project which they have designed and developed, their charges for small modifications and amendments depend on the contract they have for that particular project. If you have an existing database system which requires a minor redesign, modification, or many new features, you could pay from £1,500 and up.

New database

For a brand new database system, built around your company’s needs and customised to your requirements, a considerable amount of research into you organisation, development and programming will be required. There may be purchase costs for database software or significant work to be done in customising the system or building parts of it from the ground up. For a professionally built database solution, expect to pay around £3,000, rising depending on the scale and complexity of the project.

Bespoke solutions

If you have a unique requirement for a totally customised system, with specialist needs which require a fully programmed bespoke solution, or many legacy systems to be integrated, expect to pay anywhere from £5,000 upwards for a tailored solution.

CHOOSING A DATABASE DEVELOPER

Since databases are fundamental to so many areas of business, designing and implementing them is a huge industry. This is especially true given the amount of information that may be held by even a relatively small organisation. As such there are a large number of database developers and companies vying for your business. This is where the Approved Index, the UK's leading online business to business referral service, proves invaluable, with a list of established, proven database companies on its books for you to contact. You can complete a simple form on our website and receive up to six free quotes.

http://www.approvedindex.co.uk/indexes/DatabaseDevelopers/default.aspx

Ultimately, you will still have to choose one company to work with, so the guidance below is an overview of the process and how to choose the right supplier.

Quotations and Invitations to Tender

Getting quotes for your project couldn't be easier using Approved Index's referral system. With some high-level information about your requirements, database companies from our approved suppliers list will contact you to clarify your needs and supply you with quotes. For large or high-budget projects, it may be necessary or useful to send out a more formal Invitation to Tender, with a detailed requirements specification for the project. Companies will then respond to this with a similarly detailed quotation of how they will meet your requirements and how much they will charge. Whatever your project, it is important to find several companies to quote or tender as this will give you the most options and a better chance to find the best supplier.

Evaluate the companies

With quotations in hand, you need to choose which company to use for your project. There are several definite areas in which you can evaluate companies, though a big part of your decision should be less tangible fields like customer service and communication.

Communication and level of service

The quotation/tender process is a good chance to evaluate a company based on how well they communicate with you and the service they provide in making a quotation. Did the people you dealt with explain technical concepts to you in a way that you understand? Companies should take time to understand your requirements and should make you feel valued as a potential customer. If they don’t, it’s unlikely that you’d have a good working relationship with them.

Cost

Cost is about more than just price, and so the cheapest quote you receive will not necessarily be the best. Value is of course important, but you need to consider what you will get for you money. It’s no good choosing the cheapest supplier if you’re not confident that they’ll meet your requirements or you find communication with them difficult. A bad choice of a cheap supplier can cost you more in terms of your own time and human resources than a supplier with a higher quoted cost. Take care to strike the right balance between price and levels of service.

Experience and expertise

Clearly an established company with many years of experience is preferable to one that is just starting out, but there are many companies with a long history of satisfied customers. Experience in your industry is a definite plus as it means they will have a better understanding of any subtleties of the sector and how this relates to your database project. Equally important is experience with databases of the same scale as yours. A company which provides bespoke solutions involving small amounts of data may not be as suitable as one which is used to dealing with databases holding millions of records. Conversely, a company which builds huge databases may be less able to provide the level of customisation necessary for your project.

Stability

In this current climate, it is important to make sure that a company is solvent and has the ability to survive. You don’t want a company going bust midway through your database project.. Our recommendation is to view the latest accounts to ensure they have made no significant losses recently. It is also worth considering using Equifax or Experian for a further check.

Get references

Database developers should be happy to provide you with a list of satisfied clients as references. This is possibly the most important step in choosing the right company as talking to past clients will give you an idea what the company is like to work with in practice.

There are a number of areas you should talk to referees about. Did the company deliver what was required within budget and to schedule? Has the project stood the test of time or presented any technical problems and is it dynamic and adaptable enough that alterations to the content can be made without difficulty? Be sure to check what sort of customer service the company gives, how responsive they are to problems, and whether there were any surprise charges or costs. You should also ask them specific questions about their own database project, the supplier’s customer service and post-development availability.

Samples and prototypes

Ask suppliers for a sample database to try out. Even if it is very different to the database you want, it will still give you an idea of the quality of work the company produces. They may be willing to build a prototype of your database, but should at least provide a technical, functional specification for it. Although a prototype is by its nature incomplete and will only have a small, test dataset, it will still help you to see what the company’s work is like and if they understand your requirements correctly. A company’s willingness to provide a prototype also demonstrates that they take your project seriously.

Extras

There will almost inevitably be work and costs associated you’re your database project beyond construction of the system itself. Although it may not be strictly necessary, some level of training with the system for your staff can be invaluable. Also be clear about what support and maintenance services are included after development has been included. Particularly with bespoke solutions, you are to a degree tied to your supplier for the lifespan of the system, so check what charges they will make and be sure what services you are entitled to. Talking to past clients can help you find out what the supplier's provision is like once development is complete.

Completing the project

Once you have chosen a supplier (or maybe as part of your decision making) you need to ensure the development of your database commences and is completed, meeting your requirements and with as few problems as possible.

The course of the development should be planned out in detail. Rather than a single deadline, it is better to include a number of deadlines as milestones in the project, starting with the core functionalities and requirements and ending with the least important. This means that you can keep track of progress and catch any potential problems before they disrupt the project as a whole. It is a good idea to have your payment terms linked to these milestones. This provides an incentive to the developers and means that even in the worst case you will only pay for working parts of the system.

GLOSSARY

Column	A field, column or attribute is one type of information stored for each entity or row in a table. For example, a table storing information about people might have a ‘name’ field.
Data entry	The process of adding information to a database. This is usually done by typing it into data entry forms.
Database	A database is a system which stores similar and related information on a computer system.
DBMS	A Database Management System is a program for running and managing databases. The term is often used as a synonym for database.
Field	A field, column or attribute is one type of information stored for each entity or row in a table. For example, a table storing information about people might have a ‘name’ field.
Flat file	A flat file is a plain text file used for storing data. Comma Separated Value (CSV) files are flat files which store data in rows and columns, where each new line of the text file is a row, and the columns or fields of each row are delimited by commas.

Foreign key	A column in a table which stores the value of a primary key in another, related table. While primary keys must be unique values, foreign keys may be duplicated.
Normalisation	Normalisation is a process whereby data in a database is reorganised to remove duplication and inconsistencies.
Primary key	A column or field that uniquely identifies a row in a table. Typically, an ID column is included in a table containing unique numbers as the primary key.
Query	A query is a way of asking a question of a database, for example “which customers have placed more than one order”.
RDBMS	Relational Database Management System. A DBMS for a relational database made up of tables.
Record	A record or row is a set of information about a single entity.
Relational database	A database made up of tables. Tables are typically linked with keys defining which rows in one table ‘belong’ to the rows in another.
Report	A list or summary of information in a database on screen or for print.

Row	A row is the set of fields or columns for one entity in a table. For example, a customers table might have columns for name and address, with each row consisting of the name and address of one customer.
SQL	Structured Query Language is the most common language used to write queries for databases.
Table	A table is a store of related information, consisting of columns (or fields) of information and a number of rows, each of which is the set of fields of information for one entity. For example, a customers table might have columns for name and address, with each row consisting of the name and address of one customer.