Database types

Overview

Database types: Relational database

Without getting into too much detail, a relational-database contains tables that hold rows and columns. Each row holds related data about a particular entity (person, vehicle, sandwich, or whatever). Each column represents a piece of data about that entity (name, street address, number of pickles, and so forth).

Sometimes a piece of data naturally has more than one value. For example, a single customer might place lots of orders. To make it easy to add multiple values, those values are stored in a separate table linked to the first by some value that the corresponding records share.

Database types: Relational database

For example,

suppose you build a relational database to track your favorite street luge racers. The Racers table stores information about individual racers. Each row corresponds to a particular racer. The columns represent basic information for a racer such as name, age, height, weight, and so forth. A very important column stores each racer’s ID number.

Database types: Relational database

Over time, each racer will have lots of race results (although there will probably be lots of blank spots for bale chuckers — see www.skateluge.com/lugetalk.htm). You store race results in a separate RaceResults table. Each row records the final standings for a single racer in a single race. The columns record the racer's ID number, the race's name and date, the racer's finishing position, and the points that position is worth for overall ranking.

To find all of the finishing positions and points for a particular racer, you look up the racer's row in the Racers table, find the racer's ID number, and then find all of the rows in the RaceResults table that have this racer ID.

Racers Table		
RacerName	Nationality	RacerId
Michael Serek	Austria	1
Chris McBride	United States	2
Sebastien Tournissac	France	3

RaceResults Table					
RaceName	Division	Dates	RacerId	FinishingPosition	Points
Go Fast Speed Days	Pro Classic Luge Mass	9/1/2007-9/2/2007	1	1	450.0024
Rock and Roll	Pro Classic Luge Mass	7/27/2007-7/28/2007	1	1	450.0024
Almabtrieb World Championships	Pro Classic Luge Mass	7/11/2007-7/14/2007	1	6	403.3633
Almabtrieb World Championships	Pro Classic Luge Mass	7/11/2007-7/14/2007	2	24	321.1366
Almabtrieb World Championships	Pro Classic Luge Mass	7/11/2007-7/14/2007	3	2	432.6154
Go Fast Speed Days	Pro Classic Luge Mass	9/1/2007-9/2/2007	2	2	432.6154
Go Fast Speed Days	Pro Classic Luge Mass	9/1/2007-9/2/2007	3	3	424.3687
Top Challenge	Pro Street Luge Mass	8/25/2007-8/26/2007	2	13	0

Database types: Relational database

Relational databases have been around for a long time. (Edgar Codd started laying the foundations in 1970.) They are the most commonly used kind of database today and have been for years, so a lot of very powerful companies have spent a huge amount of time building them. All of that means that relational databases have been thoroughly studied and have evolved over time to the point where they are quite useful and effective.

Relational databases provide a number of features that make working with databases such as the street luge database easier. Some of the features they provide include:

Database types: Relational database

Data types: Each column has a particular data type (text, numeric, date, and so forth) and the database will not allow values of other types in a column.

Basic constraints: The database can enforce constraints such as requiring that a luge racer’s top speed be between 50 and 250mph (no one with a top speed less than 50 is worth recording) or it can require certain fields.

Database types: Relational database

Referential integrity: The database can prevent you from adding a RaceResults record for a racer who doesn’t exist in the Racers table. Similarly, the database can prevent you from modifying a racer’s ID if that would leave rows in the RaceResults table with invalid racer IDs, and it could prevent you from modifying a RaceResults row’s racer ID to an invalid value.

Cascading deletes and updates: If you delete a racer from the Racers table, the database can automatically delete all of that racer’s RaceResults records. Similarly if you change a racer’s ID number, the database can update the ID numbers in that racer’s RaceResults records.

Database types: Relational database

Joins: The database can quickly gather related records from different tables. For example, it can easily list every racer with his or her corresponding finishing positions sorted alphabetically and by race date.

Complex queries: Relational databases support all sorts of interesting query and aggregation functions such as SUM, AVG, MIN, COUNT, STDEV, and GROUP BY.

Database types: Relational database

Relational databases work well if:

You need to perform complicated queries and joins among different tables.

You need to perform data validations such as verifying that related rows in other tables exist.

You need to allow for any number of values for a particular piece of data (for example, race finishing positions).

You want to be able to flexibly build new queries that you didn’t plan when you started designing the project.

Database types: Relational database

Relational databases don’t work well if:

You need to use a special data topology to perform the application’s main function. For example, you can beat a hierarchy or network with a brick until it fits in a relational database but you may get better performance using a more specialized type of database.

Unless you have special needs, relational databases are usually an excellent choice.

Database types: XML

XML (eXtensible Markup Language) is a language for storing hierarchical data. XML itself doesn’t provide any tools for building, searching, updating, validating, or otherwise manipulating data and anyonewho tells you otherwise is trying to sell you something.

XML Basics: An XML file is a relatively simple text file that uses special tokens to define a structure for the data that it contains. People often compare XML to the Web language HTML (HyperText Markup Language) because both use tokens surrounded by pointy brackets, but the two languages have several large differences.

Database types: XML

XML Structures: In practice I typically see XML files used most often in one of three ways.

First, XML files are hierarchical so it’s natural to use them to hold hierarchical data. It’s straightforward to map purely hierarchical data such as a simple family tree or organizational chart into an XML file.

Second, XML files are often used to hold table-like data. The basic structure closely follows the structure of a relational database. The root element holds several table elements. Each of those elements holds “records” that hold “fields.’

Database types: XML

For example, the following XML document holds data about a simple company’s customers and their orders:

```
<AllData>
    <Customers>
        <Customer ID="1">
            <FirstName>Alfred</FirstName>
            <LastName>Gusenbauer</LastName>
        </Customer>
        <Customer ID="2">
            <FirstName>David</FirstName>
            <LastName>Thompson</LastName>
        </Customer>
        <Customer ID="3">
            <FirstName>Alberto</FirstName>
            <LastName>Selva</LastName>
        </Customer>
        </Customers>
```

```
        <Products>
            <Product ID="273645" Description="Toothbrush"
        Price="$1.95" />
            <Product ID="78463" Description="Pencil"
        Price="$0.15" />
            <Product ID="48937" Description="Notepad"
        Price="$0.75" />
        </Products>
        <CustomerOrders>
            <CustomerOrder Date="12/27/2008"
        CustomerId="2">
            <Item ID="1" ProductId="78463" Quantity="12" />
            <Item ID="2" ProductId="48937" Quantity="2" />
        </CustomerOrder>
        </CustomerOrders>
</AllData>
```

Database types: XML

The file starts with an AllData root element. That element contains three more elements that define table-like structures holding customer, product, and customer order information.

Each of these “tables” defines “records.” For example, the Customers element includes Customer “records” that hold FirstName and LastName values.

This XML document uses ID numbers to link records in different “tables” together. In this example, the single CustomerOrder element represents an order placed by customer 2 (David Thompson) who ordered 12 items with ID 78463 (pencils) and 2 items with ID 48937 (notepads).

Database types: XML

The third XML file structure I've seen regularly is a simple list of values. The following XML document uses this structure to hold configuration settings for an application:

```
<Settings>
        <NormalColor>Black</NormalColor>
        <WarningColor>Green</WarningColor>
        <ErrorColor>Yellow</ErrorColor>
        <PanicSound>panic.wav</PanicSound>
        <BugEmail>bugs@panic.com</BugEmail>
</Settings>
```

This kind of XML file gives a little more structure than a flat text file used to hold settings and lets a program use XML tools to easily load and read setting values.

Database types: XML

XML files work well if:

1. The data is naturally hierarchical.
2. Available XML tools provide the features you need.
3. You want the kinds of validation that schema files can provide.
4. You want to import and export the data in products that understand XML.

Database types: XML

XML files don’t work well if:

1. You use non-hierarchical data such as networks (described in the following section).
2. You need more complex data validation than schema files can provide.
3. You need to perform relational rather than hierarchical queries.
4. The database is very large so rewriting the entire file to update a small bit of data in the middle is cumbersome.
5. You need to allow multiple users to frequently update the database without interfering with each other.

Database types: XML

You can find lots of free tutorials covering XML and its related technologies such as XSL, XSLT, XPath, XQuery, and others on the W3 Schools Web site (www.w3schools.com).

Database types: Object-relational

An object-relational database (ORD) or object-relational database management system (ORDBMS) is a relational database that provides extra features for integrating object types into the data. Like a relational database, it can perform complex queries relatively quickly. Like an object database, it uses some special syntax to simplify the creation of objects.

Over time, many of the features originally designed for use by object-relational databases have been added to relational databases.

Database types: Object-relational

Object-relational databases and object-relational mappings work well if:

1. Your programming environment and architecture favors using objects.
2. You need to perform complicated relational-style queries.
3. You need to perform relational-style data validations.
4. Your program needs to interact with external tools where storing the data in a common relational format is an advantage.
5. You have separate programmers and database developers so maintaining a strict separation can make the project more manageable.

Database types: Object-relational

Object-relational databases and object-relational mappings don’t work well if:

You aren’t using an object-oriented language (for example, if a Microsoft Access database can do everything you need without any programming).