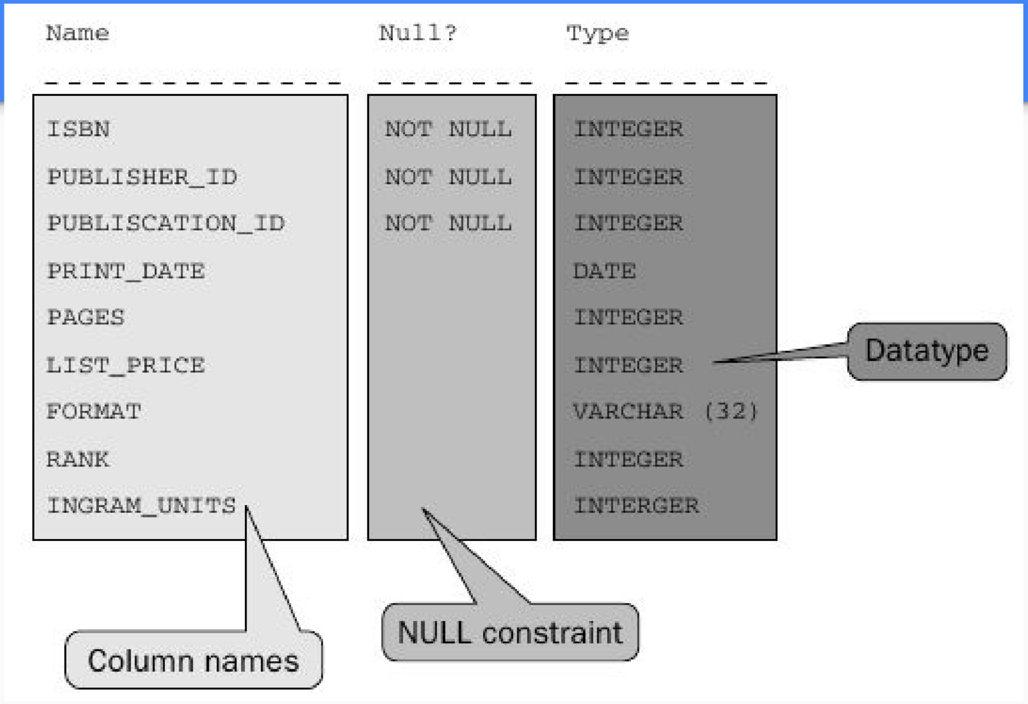
Datatypes

The terms field, column, and attribute all mean the same thing. A field applies structure and definition to a chunk of data within each repeated record.Data is not actually repeated on every record, but the structure of fields is applied to each record. So, data on each record can be different, both for the record as a whole, and for each field value. Note the use of the term “can be” rather than “is,” implying that there can be duplication across both fields and records, depending on requirements and constraints. A constraint constraints (restricts) a value. For example, in next figure the second box showing NOT NULL for the first three fields specifies that the ISBN, PUBLISHER_ID, and PUBLICATION_ID fields can never contain NULL values in any record.

Datatypes

The vertical structure of a table showing fields, constraints and datatypes.



```
Name                  Null?        Type
--------------        --------     ----------
ISBN                  NOT NULL     INTEGER
PUBLISHER_ID          NOT NULL     INTEGER
PUBLISCATION_ID       NOT NULL     INTEGER
PRINT_DATE                         DATE
PAGES                              INTEGER
LIST_PRICE                         INTEGER
FORMAT                             VARCHAR (32)
RANK                               INTEGER
INGRAM_UNITS                       INTERGER
```

Datatypes

Examine Figure 3-5 again. The left column shows the name of fields in the table. This particular table is used to contain separate editions of the same book. Separate editions of a single book can each be published by different publishers. Thus, the PUBLISHER_ID field is included in the EDITION table.

Also note the datatypes shown in previous Figure. Datatypes are shown as INTEGER, DATE, or VARCHAR(32). These three field types restrict values to be of certain content and format. INTEGER only allows whole numbers, all characters consisting of digits between 0 and 9, with no decimal point character. DATE only allows date entries where specific formatting may apply. If the default format is set to dd/mm/yyyy, an attempt to set a date value to 12/31/2004 will cause an error.

Datatypes

There are many different types of datatypes, which vary often more in name than anything else with respect to different database engines.

Datatypes can be divided into three separate sections:

- Simple
- Complex
- Specialized

Datatypes

- Simple datatypes — These are datatypes applying a pattern or value limitation on a single value such as a number.
- Complex datatypes — These include any datatypes bridging the gap between object and relational databases, including items such as binary objects and collection arrays.
- Specialized datatypes — These are present in more advanced relational databases catering to inherently structured data such as XML documents, spatial data, multimedia objects and even dynamically definable datatypes.

Datatypes: simple

Simple datatypes include basic validation and formatting requirements placed on to individual values.

This includes the following:

- Strings: Fixed length strings, Variable length strings
- Numbers: Integers, fixed length decimals, floating points
- Dates and times

Datatypes: simple

Strings: A string is a sequence of one or more characters. Strings can be fixed-length strings or variable-length strings:

Fixed length strings: Fixed-length strings — A fixed-length string will always store the specified length declared for the datatype. The value is padded with spaces even when the actual string value is less than the length of the datatype length. For example, the value NY in a CHAR(3)

Variable length strings: Variable-length strings — A variable-length string allows storage into a datatype as the actual length of the string, as long as a maximum limit is not exceeded. The length of the string is variable because when storing a string of length less than the width specified by the datatype, the string is not padded .

Datatypes: simple

Numbers — Numeric datatypes are often the most numerous field datatypes in many database tables. The following different numeric datatype formats are common:

> Integers — An integer is a whole number such that no decimal digits are included. Some databases allow more detailed specification using small integers and long integers, as well and standard-sized integer datatypes.

Datatypes: simple

Fixed-length decimals — A fixed-length decimal is a number, including a decimal point,where the digits on the right side of the decimal are restricted to a specified number of digits. For example, a DECIMAL(5,2) datatype will allow the values 4.1 and 4.12, but not 4.123. However, the value 4.123 might be automatically truncated or rounded to 4.12, depending on the database engine. More specifically, depending on the database engine

Floating points — A floating-point number is just as the name implies, where the decimal point “floats freely” anywhere within the number. In other words, the decimal point can appear anywhere in the number. Floating-point values can have any number

Datatypes: simple

Dates and times: Dates and times — Dates can be stored as simple dates or dates including timestamp information. In actuality, simple dates are often stored as a Julian date or some other similar numbering system. A Julian date is a time in seconds from a specified start date (such as January 1, 1960). When simple date values are set or retrieved in the database, they are subjected to a default formatting process spitting out to, for example, a dd/mm/yyyy format excluding seconds (depending on default database formatting settings, of course). A timestamp datatype displays both date and time information regardless of any default date formatting executing in the database.

Datatypes: complex

Complex datatypes encompass object datatypes. Available object datatypes vary for different relational databases. Some relational databases provide more object-relational attributes and functionality than others. Complex datatypes include any datatypes breaching the object-relational database divide including items such as binary objects, reference pointers, collection arrays and even the capacity to create user defined types. Following are some complex datatypes:

- Binary objects
- Reference pointers
- collection arrays
- User defined types

Datatypes: complex

- Binary objects — Purely binary objects were created in relational databases to help separate binary type data from regular relational database table record structures.and numbers. Storage issues can become problematic. Relational databases use many different types of underlying disk storage techniques to make the management of records in tables more efficient. A typical record in a table may occupy at most 2 KB (sometimes known as a page or block), and often much less. Even the smallest of graphic objects used in Web site applications easily exceeds the size of a record

Datatypes: complex

- Reference pointers — In the C programming language, a reference pointer is a variable containing an address on disk or in memory of whatever the programmer wants to point at. A pointer provides the advantage of not having to specify too much in advance with respect to how many bytes the pointer value occupies. Some relational databases allow the use of pointers where a pointer points to an object or file stored externally to the database, pointing from a field within a table, to the object stored outside the database.

Datatypes: complex

- Collection arrays — Some relational databases allow creation of what an object database would call a collection. A collection is a set of values repeated structurally (values are not necessarily the same) where the array is contained within another object, and can only be referenced from that object. In the case of a relational database, the containment factor is the collection being a field in the table. Collection arrays can have storage structures defined in alternative locations to table fields as for binary objects, but do not have to be as such. Collection arrays, much like program arrays, can be either fixed length or dynamic.

Datatypes: complex

- User-defined types — Some relational databases allow programmable or even on-the-fly creation of user-defined types. A user-defined type allows the creation of new types. Creation of a new type means that user-defined datatypes can be created by programmers, even using other user-defined types. It follows that fields can be created in tables where those fields have user-defined datatypes.

Datatypes: specialized

Specialized datatypes take into account datatypes that are intended for contained complex data objects. These specialized datatypes allow types with contained inherent structure (such as XML documents, spatial data, and multimedia objects).

Datatypes: Constraints and validation

Relational databases allow constraints, which restrict values that are allowed to be stored in table fields. Some relational databases allow the minimum of constraints necessary to define a database as being a relational database. Some relational databases allow other constraints in addition to the basics. In general, constraints are used to restrict values in tables, make validation checks on one or more fields in a table, or even check values between fields in different tables. Following are some examples of constraints:

Not null, Validation check and Keys

Datatypes: Constraints and validation

NOT NULL — This is the simplest of field level constraints, making sure that a value must always be entered into a field when a record is added or changed.

Validation check — Similar to a NOT NULL constraint, a validation checking type of constraint restricts values in fields when a record is added or changed in a table. A check validation constraint can be as simple as making sure a field allowing only M for Male or F for Female, will only ever contain those two possible values. Otherwise, check validation constraints can become fairly complex in some relational databases, perhaps allowing inclusion of user written functions running SQL scripting.

Datatypes: Constraints and validation

Keys — Key constraints include primary keys, foreign keys, and unique keys. All these key types are discussed before on in this chapter. Key constraints allow the checking and validation of values between fields in different tables.

Some relational databases allow constraints to be specified at both the field level or for an entire table as a whole, depending on the type of constraint.