



PostgreSQL Database Limits

When we use a database to store information, we are tempted to ignore the fact that on no platform do we have the luxury of infinite storage. All database systems are limited in some way, and PostgreSQL is no exception. The amount of data that can be stored in a single column, the maximum number of columns allowed in a table, and the total size of any table all have limits, albeit quite large ones.

As a limit is approached, the performance of the database will degrade. If we are, for example, manipulating very large fields consuming a large fraction of available (virtual) memory, it is likely that performance will begin to be unacceptable. Finally, PostgreSQL will be physically unable to perform an update.

Recent releases of PostgreSQL have seen most database limits relaxed, and in many cases, effectively removed. In this appendix, we will mention some of the restrictions that remain as of PostgreSQL version 8.0. For updates on limits for later versions, check out the PostgreSQL web site at <http://www.postgresql.org>.

Note The information here is derived from the PostgreSQL FAQ and mailing list contributions made by PostgreSQL developers.

Where a size is given as “No Limit,” this means that PostgreSQL alone imposes no limit. The maximum size will be determined by other factors, such as operating system limits and the amount of available disk space or virtual memory. The network transport may also impose limits. For example, there are typically limits on the size of a query that can be made via ODBC, depending on the driver. Memory limits may prevent very large columns, rows, or result sets from being created, transferred across a network (which in itself will be slow), or received by the client.

Database Size: No Limit

PostgreSQL does not impose a limit on the total size of a database. Databases of 4 terabytes (TB) are reported to exist. A database of this size is more than sufficient for all but the most demanding applications.

Due to the way that PostgreSQL arranges its data storage, you may see some performance degradation associated with databases containing many tables. PostgreSQL may use a large number of files for storing the table data, and performance may suffer if the operating system does not cope well with many files in a single directory. The introduction of tablespaces in PostgreSQL 8.0 helps the database administrator to minimize these effects. Tablespaces are covered in Chapter 11.

Table Size: 16TB–64TB

PostgreSQL normally stores its table data in chunks of 8KB. The number of these blocks is limited to a 32-bit signed integer (just over two billion), giving a maximum table size of 16TB. The basic block size can be increased when PostgreSQL is built, up to a maximum of 32KB, thereby giving a theoretical table size limit of 64TB.

Some operating systems impose a file size limit that prevent files of this size from being created, so PostgreSQL stores table data in multiple files, each 1GB in size. For large tables, this will result in many files and potential operating system performance degradation, as noted earlier.

Rows in a Table: No Limit

PostgreSQL does not impose a limit on the number of rows in any table.

Table Indexes: No Limit

There is no PostgreSQL-imposed limit on the number of indexes you can create on a table. Of course, performance may degrade if you choose to create more and more indexes on a table with more and more columns.

Field Size: 1GB

PostgreSQL has a limit of 1GB for the size of any one field in a table. In practice, the limit comes from the amount of memory available for the server to manipulate the data and transfer it to the client.

Columns in a Table: 250+

The maximum number of columns that can be accommodated in a PostgreSQL table depends on the configured block size and the type of the column. For the default block size of 8KB, at least 250 columns can be stored. This can rise to 1,600 columns if all of the columns are very simple fields, such as integer values. Increasing the block size increases these limits accordingly.

Row Size: No Limit

There is no explicit maximum size of a row. But, of course, the size of columns and their number are limited as described in the preceding text.



PostgreSQL Data Types

PostgreSQL has a particularly rich set of data types, which are described in Chapter 11 of this book, as well as in Chapter 8 of the PostgreSQL documentation.

In this appendix, we list the more useful types, ignoring some of the very specialized types and those types used only internally by PostgreSQL. Use `\dT` (or `\dT+` for even more detail) from `psql` for a definitive list of types.

In the tables in this appendix, the standard SQL name appears first, which PostgreSQL generally accepts, followed by any PostgreSQL-specific alternative names. Some types are specific to PostgreSQL and in such cases, no SQL standard name is given in the tables.

As long as it's practical, we suggest that you stick to the standard SQL types and names. Some of the official SQL names are almost invariably shortened in real use; for example `int` for integer, `bool` for boolean, and `varchar` for character varying. We have adhered to the common convention of using the shorter name in this book.

Logical Types

Table B-1 shows the PostgreSQL logical data type: `boolean`. Note that `boolean` was not officially added to the SQL language until the SQL99 standard, although it was in common use long before that.

Table B-1. *PostgreSQL Logical Data Type*

SQL Name	PostgreSQL Alternative Name	Notes
<code>boolean</code>	<code>bool</code>	Holds a truth value. Will accept values such as <code>TRUE</code> , <code>'t'</code> , <code>'true'</code> , <code>'y'</code> , <code>'yes'</code> , and <code>'1'</code> as true. Uses 1 byte of storage, and can store <code>NULL</code> , unlike a few proprietary databases.

Exact Number Types

Table B-2 shows the PostgreSQL exact number data types.

Table B-2. *postgresql Exact Number Types*

SQL Name	PostgreSQL Alternative Name	Notes
smallint	int2	A signed 2-byte integer that can store -32768 to +32767.
integer, int	int4	A signed 4-byte integer that can store -2147483648 to +2147483647.
bigint	int8	A signed 8-byte integer, giving approximately 18 digits of precision.
bit	bit	Stores a single bit, 0 or 1. To insert into a table, use syntax such as <code>INSERT INTO ... VALUES(B'1');</code>
bit varying	varbit(n)	Stores a string of bits. To insert into a table, use syntax such as <code>INSERT INTO ... VALUES(B'011101');</code>

Approximate Number Types

Table B-3 shows the PostgreSQL approximate number data types. Note that the decimal type is just an alias for numeric, which is the term used by the SQL standard and generally preferred. Similarly, rather than float, use real or double precision.

Table B-3. *PostgreSQL Approximate Number Types*

SQL Name	PostgreSQL Alternative Name	Notes
numeric (precision, scale)		Stores an exact number to the precision specified. The user guide states there is no limit to the precision that may be specified.
real	float4	A 4-byte, single-precision, floating-point number.
double precision	float8	An 8-byte, double-precision, floating-point number.
money		Equivalent to <code>numeric(9,2)</code> , storing 4 bytes of data. Its use is discouraged, as it is deprecated and support may be dropped in the future.

Temporal Types

Table B-4 shows the PostgreSQL data types for date and time.

Table B-4. *PostgreSQL Types for Date and Time*

SQL Name	PostgreSQL Alternative Name	Notes
timestamp	datetime	Stores dates and times from 4713 BC to 1465001 AD, with a resolution of 1 microsecond. You may also see <code>timestampz</code> used sometimes in PostgreSQL, which is a shorthand for <code>timestamp</code> with time zone.
interval	interval	Stores an interval of approximately $\pm 178,000,000$ years, with a resolution of 1 microsecond.
date	date	Stores dates from 4713 BC to 32767 AD, with a resolution of 1 day.
time	time	Stores a time of day, from 0 to 23:59:59.99, with a resolution of 1 microsecond.

Character Types

Table B-5 shows the PostgreSQL character data types.

Table B-5. *PostgreSQL Character Types*

SQL Name	PostgreSQL Alternative Name	Notes
char, character	bpchar	Stores a single character.
char(n)	bpchar(n)	Stores exactly n characters, which will be padded with blanks if fewer characters are actually stored.
character varying(n)	varchar(n)	Stores a variable number of characters, up to a maximum of n characters, which are not padded with blanks. This is the standard choice for character strings.
	text	A PostgreSQL-specific variant of <code>varchar</code> , which does not require you to specify an upper limit on the number of characters.

Geometric Types

Table B-6 shows the PostgreSQL geometric data types. These are specific to PostgreSQL, so there are no SQL names listed.

Table B-6. *PostgreSQL Geometric Types*

PostgreSQL Name	Notes
point	An x,y value
line	A line (pt1, pt2)
lseg	A line segment (pt1, pt2)
box	A box specified by a pair of points
path	A sequence of points, which may be closed or open
polygon	A sequence of points, effectively a closed path
circle	A point and a length, which specify a circle

Miscellaneous PostgreSQL Types

As shown in Table B-7, PostgreSQL has some other data types, which do not fit into the previous categories. SQL names are not applicable to these types.

Note that PostgreSQL does not implement the `serial` type as a separate type, although it accepts the conventional SQL syntax. Internally PostgreSQL uses an integer to store the value and a sequence to manage the automatic incrementing of the value. When a table is created with a `serial` type, an implicit sequence (named using an underscore separated combination of the table name, the column name, and `seq`) is created to manage the `serial` data column. This implicit sequence will be dropped automatically if the table is dropped.

The `cidr` type refers to Classless Inter-Domain Routing (CIDR). This is a newer standard for IP addressing. This is in contrast to the original form of IP address assignment, which uses three classes—A, B, and C—that have a network part of 8, 16, and 24 bits, respectively, allowing 16.7 million, 65 thousand, and 254 hosts per network, respectively. CIDR allows network masks of any size, so you can better allocate IP addresses and route between them in a hierarchical fashion.

Table B-7. *Other PostgreSQL Types*

PostgreSQL Name	Notes
serial	In conventional SQL usage, a <code>serial</code> (or auto-incrementing integer) is a numeric column in a table that increases each time a row is added.
oid	An object identifier. Internally, PostgreSQL adds, by default, a hidden <code>oid</code> to each row, and stores a 4-byte integer, giving a maximum value of approximately 4 billion. This type is also used as a reference when storing binary large objects. We recommend you do not use this type or rely on its existence.
cidr	Stores a network address of the form <code>x.x.x.x/y</code> where <code>y</code> is the netmask.
inet	Similar to <code>cidr</code> , except the host part can be 0.
macaddr	A MAC address of the form <code>XX:XX:XX:XX:XX:XX</code> .



PostgreSQL SQL Syntax Reference

This appendix presents a list of the PostgreSQL commands, followed by the syntax for each of these commands. This set of commands is taken from the `psql` command-line tool. Using `psql`, you can generate the complete list of commands by using the `\help` command. For the syntax of a specific command, use `\help <command>`.

More detailed explanations are available in Part VI (Reference), Section I (SQL Commands) of the PostgreSQL manual.

PostgreSQL SQL Commands

ABORT	CREATE INDEX	DROP TYPE
ALTER AGGREGATE	CREATE LANGUAGE	DROP USER
ALTER CONVERSION	CREATE OPERATOR CLASS	DROP VIEW
ALTER DATABASE	CREATE OPERATOR	END
ALTER DOMAIN	CREATE RULE	EXECUTE
ALTER FUNCTION	CREATE SCHEMA	EXPLAIN
ALTER GROUP	CREATE SEQUENCE	FETCH
ALTER INDEX	CREATE TABLE	GRANT
ALTER LANGUAGE	CREATE TABLE AS	INSERT
ALTER OPERATOR CLASS	CREATE TABLESPACE	LISTEN
ALTER OPERATOR	CREATE TRIGGER	LOAD
ALTER SCHEMA	CREATE TYPE	LOCK
ALTER SEQUENCE	CREATE USER	MOVE
ALTER TABLE	CREATE VIEW	NOTIFY
ALTER TABLESPACE	DEALLOCATE	PREPARE
ALTER TRIGGER	DECLARE	REINDEX

ALTER TYPE	DELETE	RELEASE SAVEPOINT
ALTER USER	DROP AGGREGATE	RESET
ANALYZE	DROP CAST	REVOKE
BEGIN	DROP CONVERSION	ROLLBACK
CHECKPOINT	DROP DATABASE	ROLLBACK TO SAVEPOINT
CLOSE	DROP DOMAIN	SAVEPOINT
CLUSTER	DROP FUNCTION	SELECT
COMMENT	DROP GROUP	SELECT INTO
COMMIT	DROP INDEX	SET
COPY	DROP LANGUAGE	SET CONSTRAINTS
CREATE AGGREGATE	DROP OPERATOR	SET SESSION AUTHORIZATION
CREATE CAST	DROP OPERATOR CLASS	SET TRANSACTION
CREATE CONSTRAINT TRIGGER	DROP RULE	SHOW
CREATE CONVERSION	DROP SCHEMA	START TRANSACTION
CREATE DATABASE	DROP SEQUENCE	TRUNCATE
CREATE DOMAIN	DROP TABLE	UNLISTEN
CREATE FUNCTION	DROP TABLESPACE	UPDATE
CREATE GROUP	DROP TRIGGER	VACUUM

PostgreSQL SQL Syntax

ABORT

Abort the current transaction.

```
ABORT [ WORK | TRANSACTION ]
```

ALTER AGGREGATE

Change the definition of an aggregate function.

```
ALTER AGGREGATE name ( type ) RENAME TO new_name
ALTER AGGREGATE name ( type ) OWNER TO new_owner
```

ALTER CONVERSION

Change the definition of a conversion.

```
ALTER CONVERSION name RENAME TO new_name
ALTER CONVERSION name OWNER TO new_owner
```

ALTER DATABASE

Change a database.

```
ALTER DATABASE name SET parameter { TO | = } { value | DEFAULT }
ALTER DATABASE name RESET parameter
ALTER DATABASE name RENAME TO new_name
ALTER DATABASE name OWNER TO new_owner
```

ALTER DOMAIN

Change the definition of a domain.

```
ALTER DOMAIN name
    { SET DEFAULT expression | DROP DEFAULT }
ALTER DOMAIN name
    { SET | DROP } NOT NULL
ALTER DOMAIN name
    ADD domain_constraint
ALTER DOMAIN name
    DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
ALTER DOMAIN name
    OWNER TO new_owner
```

ALTER FUNCTION

Change the definition of a function.

```
ALTER FUNCTION name ( [ type [, ...] ] ) RENAME TO new_name
ALTER FUNCTION name ( [ type [, ...] ] ) OWNER TO new_owner
```

ALTER GROUP

Change a user group.

```
ALTER GROUP groupname ADD USER username [, ... ]
ALTER GROUP groupname DROP USER username [, ... ]
ALTER GROUP groupname RENAME TO new_name
```

ALTER INDEX

Change the definition of an index.

```
ALTER INDEX name
    action [, ... ]
ALTER INDEX name
    RENAME TO new_name
```

Where action is one of:

```
OWNER TO new_owner
SET TABLESPACE indexspace_name
```

ALTER LANGUAGE

Change the definition of a procedural language.

```
ALTER LANGUAGE name RENAME TO new_name
```

ALTER OPERATOR

Change the definition of an operator.

```
ALTER OPERATOR name ( { lefttype | NONE } , { righttype | NONE } )
    OWNER TO new_owner
```

ALTER OPERATOR CLASS

Change the definition of an operator class.

```
ALTER OPERATOR CLASS name USING index_method RENAME TO new_name
ALTER OPERATOR CLASS name USING index_method OWNER TO new_owner
```

ALTER SCHEMA

Change the definition of a schema.

```
ALTER SCHEMA name RENAME TO new_name
ALTER SCHEMA name OWNER TO new_owner
```

ALTER SEQUENCE

Change the definition of a sequence generator.

```
ALTER SEQUENCE name [ INCREMENT [ BY ] increment ]
    [ MINVALUE minvalue | NO MINVALUE ]
    [ MAXVALUE maxvalue | NO MAXVALUE ]
    [ RESTART [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]
```

ALTER TABLE

Change the definition of a table.

```
ALTER TABLE [ ONLY ] name [ * ]
    action [, ... ]
ALTER TABLE [ ONLY ] name [ * ]
    RENAME [ COLUMN ] column TO new_column
ALTER TABLE name
    RENAME TO new_name
```

Where action is one of:

```

ADD [ COLUMN ] column_type [ column_constraint [ ... ] ]
DROP [ COLUMN ] column [ RESTRICT | CASCADE ]
ALTER [ COLUMN ] column TYPE type [ USING expression ]
ALTER [ COLUMN ] column SET DEFAULT expression
ALTER [ COLUMN ] column DROP DEFAULT
ALTER [ COLUMN ] column { SET | DROP } NOT NULL
ALTER [ COLUMN ] column SET STATISTICS integer
ALTER [ COLUMN ] column SET STORAGE { PLAIN | EXTERNAL | EXTENDED | MAIN }
ADD table_constraint
DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
CLUSTER ON index_name
SET WITHOUT CLUSTER
SET WITHOUT OIDS
OWNER TO new_owner
SET TABLESPACE tablespace_name

```

ALTER TABLESPACE

Change the definition of a tablespace.

```

ALTER TABLESPACE name RENAME TO new_name
ALTER TABLESPACE name OWNER TO new_owner

```

ALTER TRIGGER

Change the definition of a trigger.

```

ALTER TRIGGER name ON table RENAME TO new_name

```

ALTER TYPE

Change the definition of a type.

```

ALTER TYPE name OWNER TO new_owner

```

ALTER USER

Change a database user account.

```

ALTER USER name [ [ WITH ] option [ ... ] ]
ALTER USER name RENAME TO new_name
ALTER USER name SET parameter { TO | = } { value | DEFAULT }
ALTER USER name RESET parameter

```

Where option can be:

```

[ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| VALID UNTIL 'abstime'

```

ANALYZE

Collect statistics about a database.

```
ANALYZE [ VERBOSE ] [ table [ (column [, ...] ) ] ]
```

BEGIN

Start a transaction block.

```
BEGIN [ WORK | TRANSACTION ] [ transaction_mode [, ...] ]
```

Where `transaction_mode` is one of:

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED
                  | READ UNCOMMITTED }
READ WRITE | READ ONLY
```

CHECKPOINT

Force a transaction log checkpoint.

```
CHECKPOINT
```

CLOSE

Close a cursor.

```
CLOSE name
```

CLUSTER

Cluster a table according to an index.

```
CLUSTER index_name ON table_name
CLUSTER table_name
CLUSTER
```

COMMENT

Define or change the comment of an object.

```
COMMENT ON
{
  TABLE object_name |
  COLUMN table_name.column_name |
  AGGREGATE agg_name (agg_type) |
  CAST (source_type AS target_type) |
  CONSTRAINT constraint_name ON table_name |
  CONVERSION object_name |
  DATABASE object_name |
  DOMAIN object_name |
  FUNCTION func_name (arg1_type, arg2_type, ...) |
```

```

INDEX object_name |
LARGE OBJECT large_object_oid |
OPERATOR op (left_operand_type, right_operand_type) |
OPERATOR CLASS object_name USING index_method |
[ PROCEDURAL ] LANGUAGE object_name |
RULE rule_name ON table_name |
SCHEMA object_name |
SEQUENCE object_name |
TRIGGER trigger_name ON table_name |
TYPE object_name |
VIEW object_name
} IS 'text'

```

COMMIT

Commit the current transaction.

```
COMMIT [ WORK | TRANSACTION ]
```

COPY

Copy data between a file and a table.

```

COPY table_name [ ( column [, ...] ) ]
  FROM { 'filename' | STDIN }
  [ [ WITH ]
    [ BINARY ]
    [ OIDS ]
    [ DELIMITER [ AS ] 'delimiter' ]
    [ NULL [ AS ] 'null string' ]
    [ CSV [ QUOTE [ AS ] 'quote' ]
      [ ESCAPE [ AS ] 'escape' ]
    [ FORCE NOT NULL column [, ...] ]
  ]

```

```

COPY table_name [ ( column [, ...] ) ]
  TO { 'filename' | STDOUT }
  [ [ WITH ]
    [ BINARY ]
    [ OIDS ]
    [ DELIMITER [ AS ] 'delimiter' ]
    [ NULL [ AS ] 'null string' ]
    [ CSV [ QUOTE [ AS ] 'quote' ]
      [ ESCAPE [ AS ] 'escape' ]
    [ FORCE QUOTE column [, ...] ]
  ]

```

CREATE AGGREGATE

Define a new aggregate function.

```
CREATE AGGREGATE name (
    BASETYPE = input_data_type,
    SFUNC = sfunc,
    STYPE = state_data_type
    [ , FINALFUNC = ffunc ]
    [ , INITCOND = initial_condition ]
)
```

CREATE CAST

Define a new cast.

```
CREATE CAST (source_type AS target_type)
    WITH FUNCTION func_name (arg_types)
    [ AS ASSIGNMENT | AS IMPLICIT ]
```

```
CREATE CAST (source_type AS target_type)
    WITHOUT FUNCTION
    [ AS ASSIGNMENT | AS IMPLICIT ]
```

CREATE CONSTRAINT TRIGGER

Define a new constraint trigger.

```
CREATE CONSTRAINT TRIGGER name
    AFTER events ON
    table_name constraint attributes
    FOR EACH ROW EXECUTE PROCEDURE func_name ( args )
```

CREATE CONVERSION

Define a new conversion.

```
CREATE [DEFAULT] CONVERSION name
    FOR source_encoding TO dest_encoding FROM func_name
```

CREATE DATABASE

Create a new database.

```
CREATE DATABASE name
    [ [ WITH ] [ OWNER [=] db_owner ]
    [ TEMPLATE [=] template ]
    [ ENCODING [=] encoding ]
    [ TABLESPACE [=] tablespace ] ]
```

CREATE DOMAIN

Define a new domain.

```
CREATE DOMAIN name [AS] data_type
    [ DEFAULT expression ]
    [ constraint [ ... ] ]
```

Where constraint is:

```
[ CONSTRAINT constraint_name ]
{ NOT NULL | NULL | CHECK (expression) }
```

CREATE FUNCTION

Define a new function.

```
CREATE [ OR REPLACE ] FUNCTION name ( [ [ arg_name ] arg_type [, ... ] ] )
    RETURNS ret_type
    { LANGUAGE lang_name
      | IMMUTABLE | STABLE | VOLATILE
      | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
      | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
      | AS 'definition'
      | AS 'obj_file', 'link_symbol'
    } ...
    [ WITH ( attribute [, ... ] ) ]
```

CREATE GROUP

Define a new user group.

```
CREATE GROUP name [ [ WITH ] option [ ... ] ]
```

Where option can be:

```
    SYSID gid
    | USER username [, ...]
```

CREATE INDEX

Define a new index.

```
CREATE [ UNIQUE ] INDEX name ON table [ USING method ]
    ( { column | ( expression ) } [ opclass ] [, ... ] )
    [ TABLESPACE tablespace ]
    [ WHERE predicate ]
```


CREATE LANGUAGE

Define a new procedural language.

```
CREATE [ TRUSTED ] [ PROCEDURAL ] LANGUAGE name
    HANDLER call_handler [ VALIDATOR val_function ]
```

CREATE OPERATOR

Define a new operator.

```
CREATE OPERATOR name (
    PROCEDURE = func_name
    [, LEFTARG = left_type ] [, RIGHTARG = right_type ]
    [, COMMUTATOR = com_op ] [, NEGATOR = neg_op ]
    [, RESTRICT = res_proc ] [, JOIN = join_proc ]
    [, HASHES ] [, MERGES ]
    [, SORT1 = left_sort_op ] [, SORT2 = right_sort_op ]
    [, LTCMP = less_than_op ] [, GTCMP = greater_than_op ]
)
```

CREATE OPERATOR CLASS

Define a new operator class.

```
CREATE OPERATOR CLASS name [ DEFAULT ] FOR TYPE data_type
    USING index_method AS
    { OPERATOR strategy_number operator_name [ ( op_type, op_type ) ] [ RECHECK ]
      | FUNCTION support_number func_name ( argument_type [, ... ] )
      | STORAGE storage_type
    } [, ... ]
```

CREATE RULE

Define a new rewrite rule.

```
CREATE [ OR REPLACE ] RULE name AS ON event
    TO table [ WHERE condition ]
    DO [ ALSO | INSTEAD ] { NOTHING | command | ( command ; command ... ) }
```

CREATE SCHEMA

Define a new schema.

```
CREATE SCHEMA schema_name
    [ AUTHORIZATION username ] [ schema_element [ ... ] ]
CREATE SCHEMA AUTHORIZATION username
    [ schema_element [ ... ] ]
```

CREATE SEQUENCE

Define a new sequence generator.

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE name
    [ INCREMENT [ BY ] increment ]
    [ MINVALUE minvalue | NO MINVALUE ]
    [ MAXVALUE maxvalue | NO MAXVALUE ]
    [ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]
```

CREATE TABLE

Define a new table.

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name (
    { column_name data_type [ DEFAULT default_expr ] [ column_constraint [ ... ] ]
      | table_constraint
      | LIKE parent_table [ { INCLUDING | EXCLUDING } DEFAULTS ] } [, ... ]
)
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace ]
```

Where `column_constraint` is:

```
[ CONSTRAINT constraint_name ]
{ NOT NULL |
  NULL |
  UNIQUE [ USING INDEX TABLESPACE tablespace ] |
  PRIMARY KEY [ USING INDEX TABLESPACE tablespace ] |
  CHECK (expression) |
  REFERENCES ref_table [ ( ref_column ) ]
    [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
    [ ON DELETE action ] [ ON UPDATE action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

And `table_constraint` is:

```
[ CONSTRAINT constraint_name ]
{ UNIQUE ( column_name [, ... ] ) [ USING INDEX TABLESPACE tablespace ] |
  PRIMARY KEY ( column_name [, ... ] ) [ USING INDEX TABLESPACE tablespace ] |
  CHECK ( expression ) |
  FOREIGN KEY ( column_name [, ... ] )
    REFERENCES ref_table [ ( ref_column [, ... ] ) ]
    [ MATCH FULL | MATCH PARTIAL | MATCH SIMPLE ]
    [ ON DELETE action ] [ ON UPDATE action ] }
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

CREATE TABLE AS

Define a new table from the results of a query.

```
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } ] TABLE table_name
    [ (column_name [, ...] ) ] [ [ WITH | WITHOUT ] OIDS ]
    AS query
```

CREATE TABLESPACE

Define a new tablespace.

```
CREATE TABLESPACE tablespace_name [ OWNER username ] LOCATION 'directory'
```

CREATE TRIGGER

Define a new trigger.

```
CREATE TRIGGER name { BEFORE | AFTER } { event [ OR ... ] }
    ON table [ FOR [ EACH ] { ROW | STATEMENT } ]
    EXECUTE PROCEDURE func_name ( arguments )
```

CREATE TYPE

Define a new data type.

```
CREATE TYPE name AS
    ( attribute_name data_type [, ... ] )
```

```
CREATE TYPE name (
    INPUT = input_function,
    OUTPUT = output_function
    [ , RECEIVE = receive_function ]
    [ , SEND = send_function ]
    [ , ANALYZE = analyze_function ]
    [ , INTERNALLENGTH = { internal_length | VARIABLE } ]
    [ , PASSEDBYVALUE ]
    [ , ALIGNMENT = alignment ]
    [ , STORAGE = storage ]
    [ , DEFAULT = default ]
    [ , ELEMENT = element ]
    [ , DELIMITER = delimiter ]
)
```

CREATE USER

Define a new database user account.

```
CREATE USER name [ [ WITH ] option [ ... ] ]
```

Where option can be:

```

SYSID uid
| [ ENCRYPTED | UNENCRYPTED ] PASSWORD 'password'
| CREATEDB | NOCREATEDB
| CREATEUSER | NOCREATEUSER
| IN GROUP group_name [, ...]
| VALID UNTIL 'abs_time'

```

CREATE VIEW

Define a new view.

```
CREATE [ OR REPLACE ] VIEW name [ ( column_name [, ...] ) ] AS query
```

DEALLOCATE

Deallocate a prepared statement.

```
DEALLOCATE [ PREPARE ] plan_name
```

DECLARE

Define a cursor.

```

DECLARE name [ BINARY ] [ INSENSITIVE ] [ [ NO ] SCROLL ]
        CURSOR [ { WITH | WITHOUT } HOLD ] FOR query
        [ FOR { READ ONLY | UPDATE [ OF column [, ...] ] } ]

```

DELETE

Delete rows of a table.

```
DELETE FROM [ ONLY ] table [ WHERE condition ]
```

DROP AGGREGATE

Remove an aggregate function.

```
DROP AGGREGATE name ( type ) [ CASCADE | RESTRICT ]
```

DROP CAST

Remove a cast.

```
DROP CAST (source_type AS target_type) [ CASCADE | RESTRICT ]
```

DROP CONVERSION

Remove a conversion.

```
DROP CONVERSION name [ CASCADE | RESTRICT ]
```

DROP DATABASE

Remove a database.

DROP DATABASE name

DROP DOMAIN

Remove a domain.

DROP DOMAIN name [, ...] [CASCADE | RESTRICT]

DROP FUNCTION

Remove a function.

DROP FUNCTION name ([type [, ...]]) [CASCADE | RESTRICT]

DROP GROUP

Remove a user group.

DROP GROUP name

DROP INDEX

Remove an index.

DROP INDEX name [, ...] [CASCADE | RESTRICT]

DROP LANGUAGE

Remove a procedural language.

DROP [PROCEDURAL] LANGUAGE name [CASCADE | RESTRICT]

DROP OPERATOR

Remove an operator.

DROP OPERATOR name ({ left_type | NONE } , { right_type | NONE })
[CASCADE | RESTRICT]

DROP OPERATOR CLASS

Remove an operator class.

DROP OPERATOR CLASS name USING index_method [CASCADE | RESTRICT]

DROP RULE

Remove a rewrite rule.

DROP RULE name ON relation [CASCADE | RESTRICT]

DROP SCHEMA

Remove a schema.

```
DROP SCHEMA name [, ...] [ CASCADE | RESTRICT ]
```

DROP SEQUENCE

Remove a sequence.

```
DROP SEQUENCE name [, ...] [ CASCADE | RESTRICT ]
```

DROP TABLE

Remove a table.

```
DROP TABLE name [, ...] [ CASCADE | RESTRICT ]
```

DROP TABLESPACE

Remove a tablespace.

```
DROP TABLESPACE tablespace_name
```

DROP TRIGGER

Remove a trigger.

```
DROP TRIGGER name ON table [ CASCADE | RESTRICT ]
```

DROP TYPE

Remove a data type.

```
DROP TYPE name [, ...] [ CASCADE | RESTRICT ]
```

DROP USER

Remove a database user account.

```
DROP USER name
```

DROP VIEW

Remove a view.

```
DROP VIEW name [, ...] [ CASCADE | RESTRICT ]
```

END

Commit the current transaction.

```
END [ WORK | TRANSACTION ]
```

EXECUTE

Execute a prepared statement.

```
EXECUTE plan_name [ (parameter [, ...] ) ]
```

EXPLAIN

Show the execution plan of a statement.

```
EXPLAIN [ ANALYZE ] [ VERBOSE ] statement
```

FETCH

Retrieve rows from a query using a cursor.

```
FETCH [ direction { FROM | IN } ] cursor_name
```

Where direction can be empty or one of:

```
NEXT
PRIOR
FIRST
LAST
ABSOLUTE count
RELATIVE count
count
ALL
FORWARD
FORWARD count
FORWARD ALL
BACKWARD
BACKWARD count
BACKWARD ALL
```

GRANT

Define access privileges.

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER }
[,...] | ALL [ PRIVILEGES ] }
ON [ TABLE ] table_name [, ...]
TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { { CREATE | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
ON DATABASE db_name [, ...]
TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { CREATE | ALL [ PRIVILEGES ] }
ON TABLESPACE tablespace_name [, ...]
TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON FUNCTION func_name ([type, ...]) [, ...]
    TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { USAGE | ALL [ PRIVILEGES ] }
    ON LANGUAGE lang_name [, ...]
    TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

```
GRANT { { CREATE | USAGE } [,...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { username | GROUP group_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

INSERT

Create new rows in a table.

```
INSERT INTO table [ ( column [, ...] ) ]
    { DEFAULT VALUES | VALUES ( { expression | DEFAULT } [, ...] ) | query }
```

LISTEN

Listen for a notification.

```
LISTEN name
```

LOAD

Load or reload a shared library file.

```
LOAD 'filename'
```

LOCK

Lock a table.

```
LOCK [ TABLE ] name [, ...] [ IN lock_mode MODE ] [ NOWAIT ]
```

Where `lock_mode` is one of:

```
ACCESS SHARE | ROW SHARE | ROW EXCLUSIVE | SHARE UPDATE EXCLUSIVE
| SHARE | SHARE ROW EXCLUSIVE | EXCLUSIVE | ACCESS EXCLUSIVE
```

MOVE

Position a cursor.

```
MOVE [ direction { FROM | IN } ] cursor_name
```

NOTIFY

Generate a notification.

```
NOTIFY name
```


PREPARE

Prepare a statement for execution.

```
PREPARE plan_name [ (data_type [, ...] ) ] AS statement
```

REINDEX

Rebuild indexes.

```
REINDEX { DATABASE | TABLE | INDEX } name [ FORCE ]
```

RELEASE SAVEPOINT

Destroy a previously defined savepoint.

```
RELEASE [ SAVEPOINT ] savepoint_name
```

RESET

Restore the value of a runtime parameter to the default value.

```
RESET name
```

```
RESET ALL
```

REVOKE

Remove access privileges.

```
REVOKE [ GRANT OPTION FOR ]
  { { SELECT | INSERT | UPDATE | DELETE | RULE | REFERENCES | TRIGGER }
    [,...] | ALL [ PRIVILEGES ] }
  ON [ TABLE ] table_name [, ...]
  FROM { username | GROUP group_name | PUBLIC } [, ...]
  [ CASCADE | RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
  { { CREATE | TEMPORARY | TEMP } [,...] | ALL [ PRIVILEGES ] }
  ON DATABASE db_name [, ...]
  FROM { username | GROUP group_name | PUBLIC } [, ...]
  [ CASCADE | RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
  { CREATE | ALL [ PRIVILEGES ] }
  ON TABLESPACE tablespace_name [, ...]
  FROM { username | GROUP group_name | PUBLIC } [, ...]
  [ CASCADE | RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
  { EXECUTE | ALL [ PRIVILEGES ] }
  ON FUNCTION func_name ([type, ...]) [, ...]
  FROM { username | GROUP group_name | PUBLIC } [, ...]
  [ CASCADE | RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
  { USAGE | ALL [ PRIVILEGES ] }
  ON LANGUAGE lang_name [, ...]
  FROM { username | GROUP group_name | PUBLIC } [, ...]
  [ CASCADE | RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
  { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
  ON SCHEMA schema_name [, ...]
  FROM { username | GROUP group_name | PUBLIC } [, ...]
  [ CASCADE | RESTRICT ]
```

ROLLBACK

Abort the current transaction.

```
ROLLBACK [ WORK | TRANSACTION ]
```

ROLLBACK TO SAVEPOINT

Roll back to a savepoint.

```
ROLLBACK [ WORK | TRANSACTION ] TO [ SAVEPOINT ] savepoint_name
```

SAVEPOINT

Define a new savepoint within the current transaction.

```
SAVEPOINT savepoint_name
```

SELECT

Retrieve rows from a table or view.

```

SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ AS output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start ]
    [ FOR UPDATE [ OF table_name [, ...] ] ]

```

Where `from_item` can be one of:

```

[ ONLY ] table_name [ * ] [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
function_name ( [ argument [, ...] ] )
    [ AS ] alias [ ( column_alias [, ...] | column_definition [, ...] ) ]
function_name ( [ argument [, ...] ] ) AS ( column_definition [, ...] )
from_item [ NATURAL ] join_type from_item
    [ ON join_condition | USING ( join_column [, ...] ) ]

```

SELECT INTO

Define a new table from the results of a query.

```

SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ AS output_name ] [, ...]
    INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]
    [ ORDER BY expression [ ASC | DESC | USING operator ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start ]
    [ FOR UPDATE [ OF table_name [, ...] ] ]

```

SET

Change a runtime parameter.

```

SET [ SESSION | LOCAL ] name { TO | = } { value | 'value' | DEFAULT }
SET [ SESSION | LOCAL ] TIME ZONE { time_zone | LOCAL | DEFAULT }

```

SET CONSTRAINTS

Set constraint checking modes for the current transaction.

```
SET CONSTRAINTS { ALL | name [, ...] } { DEFERRED | IMMEDIATE }
```

SET SESSION AUTHORIZATION

Set the session user identifier and the current user identifier of the current session.

```
SET [ SESSION | LOCAL ] SESSION AUTHORIZATION username
SET [ SESSION | LOCAL ] SESSION AUTHORIZATION DEFAULT
RESET SESSION AUTHORIZATION
```

SET TRANSACTION

Set the characteristics of the current transaction.

```
SET TRANSACTION transaction_mode [, ...]
SET SESSION CHARACTERISTICS AS TRANSACTION transaction_mode [, ...]
```

Where `transaction_mode` is one of:

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED
                  | READ UNCOMMITTED }
READ WRITE | READ ONLY
```

SHOW

Show the value of a runtime parameter.

```
SHOW name
SHOW ALL
```

START TRANSACTION

Start a transaction block.

```
START TRANSACTION [ transaction_mode [, ...] ]
```

Where `transaction_mode` is one of:

```
ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED
                  | READ UNCOMMITTED }
READ WRITE | READ ONLY
```

TRUNCATE

Empty a table.

```
TRUNCATE [ TABLE ] name
```

UNLISTEN

Stop listening for a notification.

```
UNLISTEN { name | * }
```

UPDATE

Update rows of a table.

```
UPDATE [ ONLY ] table SET column = { expression | DEFAULT } [, ...]  
    [ FROM from_list ]  
    [ WHERE condition ]
```

VACUUM

Garbage-collect and optionally analyze a database.

```
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] [ table ]  
VACUUM [ FULL ] [ FREEZE ] [ VERBOSE ] ANALYZE [ table [ (column [, ...] ) ] ]
```



psql Reference

This appendix defines the `psql` command-line options and internal commands. This information is taken from the `psql` command-line tool's internal help.

Command-Line Options

`psql` has the following command-line usage:

```
psql [options] [dbname [username]]
```

Table D-1 shows the command-line options.

Table D-1. psql Command-Line Options

Option	Meaning
-?, --help	Show help, then exit
-a, --echo_all	Echo all input from script
-A, --no-align	Unaligned table output mode (-P format=unaligned)
-c, --command <command>	Run only single command (SQL or internal) and exit
-d, --dbname <dbname>	Specify database name to connect to (default: current username)
-e, --echo-queries	Echo commands sent to server
-E, --echo-hidden	Display queries that internal commands generate
-f, --file <filename>	Execute commands from file, then exit
-F, --field-separator <string>	Set field separator (default: " ") (-P fieldsep=)
-h, --host <hostname>	Database server host or socket directory (default: "local socket")
-H, --html	HTML table output mode (-P format=html)
-l, --list	List available databases, then exit
-n	Disable enhanced command-line editing (readline)
-o, --output <filename>	Send query results to file (use -o program for a pipe)
-p, --port <port>	Database server port (default: 5432)

Table D-1. *psql Command-Line Options (Continued)*

Option	Meaning
-P, --pset var[=arg]	Set printing option var to arg (see \pset command)
-q, --quiet	Run quietly (no messages, only query output)
-R, --record-separator <string>	Set record separator (default: newline) (-P recordsep=)
-s, --single-step	Single-step mode (confirm each query)
-S, --single-line	Single-line mode (end of line terminates SQL command)
-t, --tuples-only	Print rows only (-P tuples_only)
-T, --table-attr <text>	Set HTML table tag attributes (width, border) (-P tableattr=)
-U, --username <name>	Database username (default: current username)
-v, --set, --variable name=value	Set psql variable name to value
-V, --version	Output version information, then exit
-W, --password	Prompt for password (should happen automatically)
-x, --expanded	Turn on expanded table output (-P expanded)
-X, --no-psqlrc	Do not read startup file (~/.psqlrc)

Internal Commands

Table D-2 lists the psql internal commands.

Table D-2. *psql Internal Commands*

Command	Meaning
\! <command>	Execute command in shell or start interactive shell
\a	Toggle between unaligned and aligned output mode
\c[onnect] [<dbname> - [<user>]]	Connect to new database
\C <string>	Set table title, or unset if none
\cd <dir>	Change the current working directory
\copy	Perform SQL COPY with data stream to the client host
\copyright	Show PostgreSQL usage and distribution terms
\d, \d+ <name>	Describe table, index, sequence, or view (with + gives expanded output)
\d{t i s v S} <pattern>	List tables/indexes/sequences/views/system tables
\da <pattern>	List aggregate functions
\db, \db+ <pattern>	List tablespaces (with + gives expanded output)

Table D-2. *psql Internal Commands (Continued)*

Command	Meaning
<code>\dc <pattern></code>	List conversions
<code>\dC</code>	List casts
<code>\dd <pattern></code>	Show comment for object
<code>\dD <pattern></code>	List domains
<code>\df, \df+ <pattern></code>	List functions (with + gives expanded output)
<code>\dg <pattern></code>	List groups
<code>\dn, \dn+ <pattern></code>	List schemas (with + gives expanded output)
<code>\do <name></code>	List operators
<code>\dl</code>	List large objects, same as <code>\lo_list</code>
<code>\dp <pattern></code>	List table, view, and sequence access privileges
<code>\dT, \dT+ <pattern></code>	List data types (with + gives expanded output)
<code>\du <pattern></code>	List users
<code>\e[dit] [<file>]</code>	Edit the query buffer (or file) with external editor
<code>\echo <string></code>	Write string to standard output
<code>\encoding [<encoding>]</code>	Show or set client encoding
<code>\f [<string>]</code>	Show or set field separator for unaligned query output
<code>\g <file></code>	Send query buffer to server (and results to file or program for a pipe)
<code>\h[elp] [<name>]</code>	Help on syntax of SQL commands; * for all commands
<code>\H</code>	Toggle HTML output mode
<code>\i <file></code>	Execute commands from file
<code>\l[ist], \l[ist]+</code>	List all databases (with + gives expanded output)
<code>\lo_export <loboid> <file></code>	Export large objects
<code>\lo_import <file> [<comment>]</code>	Import large objects
<code>\lo_list</code>	List large objects
<code>\lo_unlink <loboid></code>	Delete large objects
<code>\o <file></code>	Send all query results to file or program for a pipe
<code>\p</code>	Show the contents of the query buffer
<code>\pset name [value]</code>	Set table output option (name := {format border expanded fieldsep footer null recordsep tuples_only title tableattr pager})
<code>\q</code>	Quit psql
<code>\qecho <string></code>	Write string to query output stream (see <code>\o</code>)

Table D-2. psql Internal Commands (Continued)

Command	Meaning
<code>\r</code>	Reset (clear) the query buffer
<code>\s [<file>]</code>	Display history or save it to file
<code>\set [<name> [<value>]]</code>	Set internal variable, or list all if no parameters
<code>\t</code>	Show only rows
<code>\T [<string>]</code>	Set HTML <table> tag attributes, or unset if none
<code>\timing</code>	Toggle timing of commands
<code>\unset <name></code>	Unset (delete) internal variable
<code>\w <file></code>	Write query buffer to file
<code>\x</code>	Toggle expanded output
<code>\z <pattern></code>	List table, view, and sequence access privileges (same as <code>\dp</code>)



Database Schema and Tables

The database schema used in the examples in this book is a simplified customer/orders/items database, as shown in Figure E-1.

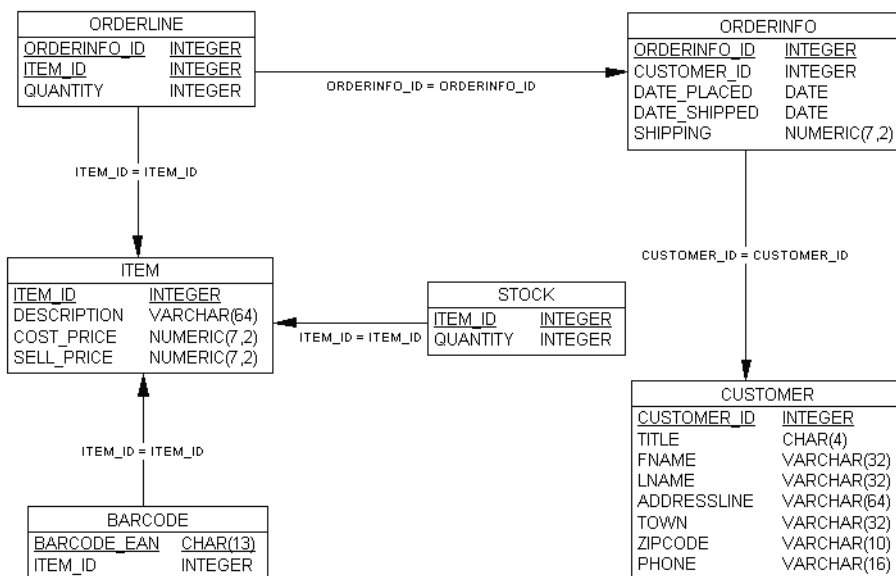


Figure E-1. Final schema design

The tables need to be created in an appropriate order so that dependent tables are created first, because of the foreign key constraints (see Chapter 8). This is the same order to be followed as data is inserted into the tables. The appropriate order is as follows:

- customer
- orderinfo
- item
- orderline
- stock
- barcode

The SQL to create the final version of this sample database, `bpfinal`, including the foreign key constraints follows. This code can be found in the download bundle available from the Downloads section of the Apress web site (<http://www.apress.com>) as `create_tables-bpfinal.sql`. The simplified version, `bpsimple` (see Chapter 3), excluding the foreign key constraints, can be found in `create_tables-bpsimple.sql`.

The download code bundle has the table-population commands, `pop-all-tables.sql`, in an appropriate order ready for populating either schema.

Customer table

```
create table customer
(
    customer_id          serial,
    title                char(4),
    fname                varchar(32),
    lname                varchar(32) not null,
    addressline          varchar(64),
    town                 varchar(32),
    zipcode               char(10) not null,
    phone                varchar(16),
    CONSTRAINT           customer_pk PRIMARY KEY(customer_id)
);
```

Orderinfo table

```
create table orderinfo
(
    orderinfo_id         serial,
    customer_id           integer not null,
    date_placed           date not null,
    date_shipped          date,
    shipping               numeric(7,2) ,
    CONSTRAINT            orderinfo_pk PRIMARY KEY(orderinfo_id),
    CONSTRAINT            orderinfo_customer_id_fk FOREIGN KEY(customer_id) REFERENCES
customer(customer_id)
);
```

Item table

```
create table item
(
    item_id               serial,
    description            varchar(64) not null,
    cost_price             numeric(7,2),
    sell_price             numeric(7,2),
    CONSTRAINT            item_pk PRIMARY KEY(item_id)
);
```

Orderline table

```
create table orderline
(
  orderinfo_id          integer not null,
  item_id               integer not null,
  quantity              integer not null,
  CONSTRAINT            orderline_pk PRIMARY KEY(orderinfo_id,
item_id),
  CONSTRAINT orderline_orderinfo_id_fk FOREIGN KEY(orderinfo_id) REFERENCES
orderinfo(orderinfo_id),
  CONSTRAINT orderline_item_id_fk FOREIGN KEY(item_id) REFERENCES item(item_id)
);
```

Stock table

```
create table stock
(
  item_id               integer not null,
  quantity              integer not null,
  CONSTRAINT            stock_pk PRIMARY KEY(item_id),
  CONSTRAINT stock_item_id_fk FOREIGN KEY(item_id) REFERENCES item(item_id)
);
```

Barcode table

```
create table barcode
(
  barcode_ean           char(13) not null,
  item_id               integer not null,
  CONSTRAINT            barcode_pk PRIMARY KEY(barcode_ean),
  CONSTRAINT barcode_item_id_fk FOREIGN KEY(item_id) REFERENCES item(item_id)
);
```



Large Objects Support in PostgreSQL

Traditionally, databases have been able to store data in a limited number of forms, usually as numeric values (integers, floating point, and fixed point) and text strings. Often, the size of the text data is limited. In the past, even PostgreSQL enforced a limit of a few thousand characters as the maximum length of a field.

It might be useful to be able to create a database application that can handle *arbitrary* unstructured data formats, such as images. As an example of how to handle large data items, we will consider how we might add photographs to a database. There are several ways we could do this:

- Use links into a file system or the Web
- Use encoded long text strings
- Use BLOBs

We will look at each of these approaches in this appendix, using the sample database we built in this book, `bpfinal` (see Chapter 8 and Appendix E for details on the `bpfinal` schema). We will see how we might add images of products, so that our web-based interface could provide an online catalog.

Using Links

Our first option is to avoid storing images in the physical database at all. The idea is to place all of the images in the normal filing system of a server, which may be the database server, a file sharing server, or a web server. The database itself then only needs to contain a text link to the file. Any client application would follow the link to retrieve the image.

We need to create an additional table in the database to store the image links. It is very similar to the `stock` table in that we are providing additional information for each item we carry:

```
CREATE TABLE image
(
    item_id    integer NOT NULL,
    picture    varchar(512),
    CONSTRAINT image_pk PRIMARY KEY(item_id),
    CONSTRAINT image_item_id_fk FOREIGN KEY(item_id)
        REFERENCES item(item_id)
);
```

Here, we have added constraints to ensure that we can add images only for items that exist.

Now we can update the `image` table with links to product photographs:

```
INSERT INTO image VALUES (3, 'http://server/images/rubik.jpg');
INSERT INTO image VALUES (9, '//server/images/coin.bmp');
INSERT INTO image VALUES (5, '/mnt/server/images/frame.png');
```

This solution has both advantages and disadvantages. Storing links rather than pictures means that the database size is kept to a minimum and applications will be portable to other database systems, as we have not used any esoteric features to handle images. Furthermore, retrieving the actual images will also be very fast, as reading a file from the file system will typically be many times faster than querying a database. Also we have a wide choice of locations for storing the images. In this example, we have used the following:

- A URL to provide a link into a web server
- A UNC file reference for a Windows file server
- A reference to an NFS-mounted UNIX server

However, by using links we are unable to enforce referential integrity in the database. If the images that are stored elsewhere are changed or deleted, the database is not automatically updated. To back up our system, we will need to attend to image files on the file system (and elsewhere), as well as the database itself. We must also ensure that the links we use are in a form that all possible clients can use. For example, the NFS form requires that all clients access the shared files in the same way and have access rights on the server.

Using Encoded Text Strings

In PostgreSQL 7.1, the limit on the size of a field was raised to 1GB. For all practical purposes, this is effectively unlimited. We could consider using the `text` type to store the images in the database directly. This is possible, if a little tricky.

Images are in general binary data, not well suited to a character type. So, we need to encode the image in some way, perhaps by using hexadecimal or MIME encoding. Handling the very long strings that result may also cause a problem with limits imposed by client applications, network transport mechanisms, or ODBC drivers. The storage space needed for encoded strings will also be up to double the size of the binary image file.

Using BLOBs

One of the wide variety of data types PostgreSQL currently supports is the binary large object, or BLOB, which is suitable for storing large data items. This allows us to create a database application that can handle *arbitrary* unstructured data formats, such as images. Thus, we can store our image data, or any large or binary data object, in a PostgreSQL database by using BLOBs.

PostgreSQL supports a column type of `oid`, which is an object identifier, a reference to arbitrary data. These are used to manage the BLOBs and can be used to transfer the contents of any file into the database, and to extract an object from the database into a file. They can therefore be used to handle our product images, or any other data that we might wish to store.

We can modify the sample `image` table definition from earlier in this appendix to use BLOBs by specifying `oid` as the image data type:

```
CREATE TABLE image
(
    item_id    integer NOT NULL,
    picture    oid,
    CONSTRAINT image_pk PRIMARY KEY(item_id),
    CONSTRAINT image_item_id_fk FOREIGN KEY(item_id) REFERENCES item(item_id)
);
```

Importing and Exporting Images

PostgreSQL provides a number of functions that can be used in SQL queries for inserting into, retrieving from, and deleting BLOB data in a table.

To add an image to the table, we can use the SQL function `lo_import`, like this:

```
INSERT INTO image VALUES (3, lo_import('/tmp/image.jpg'));
```

The contents of the specified file are read into a BLOB object and stored in the database. The `image` table will now have a non-NULL `oid` that references the BLOB:

```
bpfinal=# SELECT * FROM image;
 item_id | picture
-----+-----
        3 | 163055
(1 row)
```

```
bpfinal=#
```

We can see all of the large objects stored in the database by using a `psql` internal command, `\lo_list`, or `\dl`, to list them:

```
bpfinal=# \dl
      large objects
 ID | Description
-----+-----
163055 |
(1 row)
```

```
bpfinal=#
```

Large objects are retrieved from the database using `lo_export`, which writes a file containing the contents of the BLOB:

```
bpfinal=# SELECT lo_export(picture, '/tmp/image2.jpg')
bpfinal=# FROM image WHERE item_id = 3;
lo_export
-----
          1
(1 row)

bpfinal=#
```

We can delete a large object from the database with `lo_unlink`:

```
bpfinal=# SELECT lo_unlink(picture) FROM image WHERE item_id = 3;
lo_unlink
-----
          1
(1 row)

bpfinal=# \dl
Large objects
ID | Description
----+-----
(0 rows)

bpfinal=#
```

We must be careful when we delete large objects, as any references to the BLOB will remain:

```
bpfinal=# SELECT * FROM image;
item_id | picture
-----+-----
        3 | 163055
(1 row)

bpfinal=#
```

As operations on a large object and its object identifier are essentially decoupled, we must make sure to take care of both when manipulating BLOBs. So, when deleting a BLOB, we should set the object reference to `NULL` to prevent errors in our client applications:

```
bpfinal=# UPDATE image SET picture=NULL WHERE item_id = 3;
```

In the examples here, we have used `psql` to manipulate binary objects. It is important to understand that the import and export functions `lo_import` and `lo_export` are executed by the back-end database server, not by `psql`. Any client application using SQL statements can use these SQL functions to add and update BLOBs.

There are three caveats when importing and exporting BLOBs:

- As the import is performed by the server, the files read and written by the `import` and `export` must be specified using a path and filename that are accessible to the server, rather than the client. If in `psql` we had simply said this:

```
INSERT INTO image VALUES (3, lo_import('image.jpg'));
```

and expected PostgreSQL to insert the contents of a file in the current directory, we would have received an error message. This is because the `import` fails to find the file. We need to arrange that files for `import` are (temporarily) placed in a location that the server can access. Similarly, we need to use full filenames for exporting binary objects.

- Exported files must be placed in a directory that the server user can write; that is, a location where the operating system user `postgres` has permission to create files.
- All large object manipulation must take place within a SQL transaction—between `BEGIN` and `COMMIT` or `END` statements. By default, `psql` executes each SQL statement in its own transaction, so this is not a problem, but client applications that perform imports and exports must be written with transactions.

Remote Importing and Exporting

Because the SQL functions `lo_import` and `lo_export` use the server file system, using them can be inconvenient when creating BLOBs. However, `psql` contains internal commands that can be used to import and export binary objects from a remote client machine.

We can add a BLOB to the database by using `\lo_import` and passing it a local filename. In this case, files in the current directory will work just fine, and we can see the object listed with `\lo_list`:

```
bpfinal=# \lo_import image.jpg
lo_import 163059
bpfinal=# \lo_list
      Large objects
      ID | Description
-----+-----
 163059 |
(1 row)

bpfinal=#
```

Now we need to associate the BLOB with the `image` table by updating the appropriate row:

```
bpfinal=# UPDATE image SET picture=163059 WHERE item_id = 3;
UPDATE 1
bpfinal=# SELECT * FROM image;
 item_id | picture
-----+-----
       3 | 163059
(1 row)

bpfinal=#
```

We can extract a BLOB with `\lo_export`, specifying the required object identifier and a file to write. Again, a local filename is fine:

```
bpfinal=# \lo_export 163059 image2.jpg
lo_export
```

```
bpfinal=#
```

Finally, we can delete a large object with `\lo_unlink`:

```
bpfinal=# \lo_unlink 163059
lo_unlink 163059
```

```
bpfinal=#
```

Programming BLOBs

As you might expect, it is possible to use BLOB import and export functions from the programming languages supported by PostgreSQL.

From C, using the `libpq` library (see Chapter 13), we can use the functions `lo_import`, `lo_export`, and `lo_unlink` in much the same way as described in the preceding sections:

```
Oid lo_import(PGconn *conn, const char *filename);
int lo_export(PGconn *conn, Oid loObjId, const char *filename);
int lo_unlink(PGconn *conn, Oid loObjId);
```

Here is a sample program that imports an image file into the database. Note that the large object functions must be called within a transaction:

```
#include <stdlib.h>
#include <libpq-fe.h>

int main()
{
    PGconn *myconnection = PQconnectdb("");
    PGresult *res;
    Oid blob;

    if(PQstatus(myconnection) == CONNECTION_OK)
        printf("connection made\n");
    else
        printf("connection failed\n");

    res = PQexec(myconnection, "begin");
    PQclear(res);

    blob = lo_import(myconnection, "image.jpg");
    printf("import returned oid %d\n", blob);
```

```
res = PQexec(myconnection, "end");
PQclear(res);

PQfinish(myconnection);
return EXIT_SUCCESS;
}
```

When we compile and run the program, we can see the new binary object identifier reported. (This program is included in the sample programs for Chapter 13.)

```
$ make import
cc -I/usr/local/pgsql/include -L/usr/local/pgsql/lib -lpq import.c -o import
$ PGDATABASE=bpfinal ./import
connection made
import returned oid 163066
$
```

BLOBs can be imported and exported from other languages in similar ways.

For finer control over large object access, PostgreSQL provides a suite of low-level functions akin to open, read, write, and others for ordinary files:

```
int lo_open(PGconn *conn, Oid lojId, int mode);
int lo_close(PGconn *conn, int fd);
int lo_read(PGconn *conn, int fd, char *buf, size_t len);
int lo_write(PGconn *conn, int fd, char *buf, size_t len);
int lo_lseek(PGconn *conn, int fd, int offset, int whence);
Oid lo_creat(PGconn *conn, int mode);
int lo_tell(PGconn *conn, int fd);
```

Refer to the online documentation for more details on these functions.

INDEX

■ Symbols

- ! command (\!), psql, 121, 574
- % (percentage) character
 - pattern matching in SELECT, 91
- * (asterisk) character
 - SELECT statement, 79
- .NET Framework
 - open source implementation of, 520
- ? command (\?), psql, 78, 115, 119
- ? option (-?), psql, 573
- \ (backslash) character
 - see under* escape characters
- @ symbol
 - PHP error suppression, 458
- _ (underscore) character
 - pattern matching in SELECT, 91

■ A

- a command (\a), psql, 119, 574
- a option (-a)
 - createuser utility, 323
 - pg_dump utility, 341
 - pg_restore utility, 342
 - vacuumdb utility, 351
- A option (-A), psql, 118, 573
- a option (-a), psql, 118, 573
- ABORT command, 552
- abs function, 214, 274
- absolute method
 - ResultSet interface, java.sql, 503
- absolute value operator, 271
- acceptsURL method
 - Driver interface, java.sql, 498
- Access
 - see* Microsoft Access
- access permissions
 - listing, 121
- accessing data
 - see* data access
- accessing PostgreSQL
 - see under* PostgreSQL
- ACID rules, transactions, 246–247
- acos function, 275
- add_one function, 279
- addBatch method
 - Statement interface, java.sql, 510
- adding data to database
 - see* INSERT statement
 - see also* data handling; inserting data into database
- addition operator, 271
 - operator precedence, 270
- addresses
 - database design, 364
- administration, 309–356
 - database backup and recovery, 338–346
 - database initialization, 317–318
 - database performance, 347–356
 - PostgreSQL internal configuration, 320–338
 - server control, 318–320
 - system configuration, 309–316
- ADO.NET objects
 - relationships between, 537
- AFTER triggers, 300
- afterLast method
 - ResultSet interface, java.sql, 504
- aggregate functions, SELECT, 173–185
 - avg function, 184–185
 - count function, 174–182
 - GROUP BY clause, 174, 176–178
 - using with HAVING clause, 179
 - HAVING clause, 174, 178–181
 - max function, 183–184
 - median function, 185
 - min function, 182–183
 - mode function, 185
 - NULL values, 183
 - optional clauses, 174
 - standard deviation functions, 174
 - sum function, 184
 - table listing of, 174
 - variance functions, 174
 - WHERE clause and, 178

- aggregates
 - ALTER AGGREGATE, 552
 - CREATE AGGREGATE, 558
 - DROP AGGREGATE, 563
 - listing, 119
- aliases
 - column aliases in SELECT, 81
 - correlated subqueries, 189
 - table name aliases, 105
 - used in subquery, 187
- ALIAS declaration, 285
- aligned table output mode, `psql`
 - toggleing between unaligned and, 119
- ALL privilege
 - grantable privileges, 337
- allow rules
 - `pg_hba.conf` file, 312
- ALTER AGGREGATE command, 552
- ALTER CONVERSION command, 552
- ALTER DATABASE command, 329, 553
- ALTER DOMAIN command, 553
- ALTER FUNCTION command, 553
- ALTER GROUP command, 325, 553
- ALTER INDEX command, 553
- ALTER LANGUAGE command, 554
- ALTER OPERATOR CLASS command, 554
- ALTER OPERATOR command, 554
- ALTER SCHEMA command, 554
- ALTER SEQUENCE command, 554
- ALTER TABLE command, 224–227, 554
 - foreign key constraint, 235
- ALTER TABLESPACE command, 327, 555
- ALTER TRIGGER command, 555
- ALTER TYPE command, 555
- ALTER USER command, 323, 555
- ANALYZE command, 556
- ANALYZE option
 - EXPLAIN statement, 350
 - VACUUM command, 349
- AND (Binary AND) operator, 271
- AND conditional operator
 - choosing rows in SELECT, 88, 89
 - operator precedence, 270
 - relating tables, 102, 108
- ANSI isolation levels, transactions, 260–261
- anti-logarithm operator, 270
- APIs, data access with, 15
- applications directory
 - adding to execution path, 58
 - installing on Windows, 59
- approximate number data types, 546
- architecture, 13, 14
- arguments
 - see also* parameters
 - `ecpg`, 424
 - PL/pgSQL functions, 283
- arithmetic operators, 270–271
 - precedence, 269
 - unary arithmetic operators, 271
- array operator
 - operator precedence, 270
- arrays, 210–212
 - PostgreSQL style, 210
 - SQL99 style, 211
- AS keyword
 - changing data type, 86
 - column aliases, 81, 82
- ASC keyword
 - ORDER BY clause in SELECT, 81, 82, 83
 - default sort order, 82
- asin function, 275
- assignments
 - stored procedures, 288
 - PERFORM statement, 289
 - SELECT INTO statement, 288
- associativity of operators, 269
- asterisk (*)
 - SELECT statement using, 79
- asynchronous working
 - using `libpq`, 411–417
 - canceling queries, 415
 - executing queries, 412
 - making asynchronous database connection, 415
- atan functions, 275
- atomicity
 - ACID transaction rules, 246
 - RDBMS, 7
- audit trails, 11
- authentication
 - `pg_hba.conf` file, 311, 312
 - name mapping, 311
 - prompting for superuser password, 317
 - trust mechanism, 55
- authentication_timeout option
 - PostgreSQL.conf file, 314

- AUTHORIZATION syntax
 - CREATE SCHEMA command, 332
 - SET SESSION AUTHORIZATION, 571
- available_drivers function, Perl DBI, 483
- avg function, 184–185
 - description, 174
 - DISTINCT keyword, 185
 - NULL values, 184
 - used in subquery, 186
- B**
- B option (-B)
 - postmaster.opts file, 316
- background processes
 - running processes on Linux and UNIX, 318
- backslash (\)
 - see under* escape characters
- backups
 - creating backup, 339–341
 - database backup and recovery, 338–346
 - pgAdmin III tool, 128, 343–346
 - restoring from backup, 341–343
 - utility to back up database, 311
- barcode table
 - creating table, 68, 579
 - identifying primary keys, 372
 - populating sample database tables, 70
- base directory
 - Linux and Windows, 309, 310
 - subdirectories, 310
- batches, SQL
 - java.sql.Statement interface, 509
- BatchUpdateException class
 - JDBC API, 494
- BEFORE triggers, 300
- beforeFirst method
 - ResultSet interface, java.sql, 504
- BEGIN ... COMMIT blocks
 - ecpg programs, 424
 - transactions, 244
- BEGIN ... END blocks, 556
 - loops, 293
- BeginTransaction method
 - NpgsqlConnection class, 522
- Berkeley Software Distribution (BSD)
 - open-source software, 15
- BETWEEN keyword
 - choosing rows in SELECT, 89, 90
 - letter range comparison behavior, 90
 - operator precedence, 270
- bigint data type, 546
- bin directory
 - executable files, 310
 - PostgreSQL installation, 47
 - system configuration, 310, 311
- binary packages
 - installing PostgreSQL on Linux, 44
- binary values
 - libpq using, 411
- binding parameters
 - using Perl DBI, 481–483
- bindir option
 - configure script, 51
 - pg_config command, 52
- bit data type, 546
- BLOBs (binary large objects)
 - deleting, 584
 - importing and exporting images, 583–585
 - programming BLOBs, 586, 587
 - remote importing and exporting, 585–586
 - support for large objects, 583–587
- block comments, 284
- block-structured languages, 282
- boolean data type, 202–204, 545
- box data type, 209, 548
- bpchar data type, 547
- bpfinal
 - design, 238
 - schema, 577–579
- bpsimple database
 - creating sample database, 64
- buffers
 - psql command resetting, 78
 - setting number used, 314
 - shared memory buffers, 316
- built-in functions, 273–275
 - listing, 273
 - mathematical functions, 274
 - operator equivalents, 274
- bundles, 469
- business rules
 - implementing, 377
- C**
- C command (\C), psql, 119, 574
- c command (\c), psql, 119, 574
 - creating first database, 114
- C option (-C)
 - pg_dump utility, 341
 - pg_restore utility, 342

- c option (-c)
 - pg_dump utility, 341
 - pg_restore utility, 342
 - psql, 118, 573
- C programming language
 - accessing PostgreSQL from C using libpq, 385–417
 - creating executable program, 422
 - ecpg creating C file, 421
 - ecpg translated source code, 421
 - functions specific to PostgreSQL
 - see libpq functions
 - see also libpq library
 - writing esqlc program, 420–422
- C#
 - accessing PostgreSQL from, 517–541
 - most practical way to use, 520
 - Npgsql in Mono, 520–539
 - Npgsql in Visual Studio, 539
 - ODBC .NET data provider on Windows, 517–520
- calculations
 - performing in SELECT, 86
 - with dates and times, 100
- callable statements
 - JDBC sending SQL statements to database, 507
- CallableStatement interface, java.sql, 507
- callbacks
 - asynchronous working, libpq, 412
- cancelRowUpdates method
 - ResultSet interface, java.sql, 506
- candidate keys
 - database design, 372
- cardinality
 - cardinality symbols, 367
 - relating data entities, 366
- CASCADE keyword, 241, 242
 - DROP SCHEMA command, 333
- CASE function
 - execution control structures, 292
- case sensitivity
 - Boolean data type values, 202
 - data in SQL databases, 79
 - embedded SQL keywords, 421
 - SQL command keywords, 10, 79
- cast function, 213
 - date and time data types, 96, 97, 98
 - performing calculations in SELECT, 86
 - used in subquery, 185, 187
- CAST operator
 - operator precedence, 270
- casts
 - CREATE CAST command, 558
 - DROP CAST command, 563
 - listing, 120
- cbirt function, 274
- cd command (\cd), psql, 119, 574
- ceil function, 274
- Celko, Joe
 - normalization, 33
- cells, spreadsheets, 18
- chained mode
 - implicit transactions, 261
- ChangeDatabase method
 - NpgsqlConnection class, 522
- changing isolation level
 - transaction isolation, 261
- char data type, 40, 204, 547
- char_length function, 275
- character data types, 204–206, 547
 - binary data, 582
 - choosing between, 204
 - inserting into database, 151
- character encoding
 - PHP support for, 459
- charting
 - using Microsoft Excel, 142
- CHECK keyword
 - column constraints, 218, 220
 - table constraints, 222
- CHECKPOINT command, 556
- chmod command
 - making file executable, 58
- chown command
 - creating tablespaces, 327
- cidr data type, 209, 549
- CIDR-ADDRESS column
 - pg_hba.conf file, 312
- circle data type, 548
- Classless Inter-Domain Routing data type, 548
- ClassNotFoundException
 - implementing Driver interface, 496

- CLASSPATH
 - implementing Driver interface, 496
 - installing PostgreSQL JDBC driver, 493
- clearBatch method
 - Statement interface, java.sql, 510
- clearParameters method
 - PreparedStatement interface, java.sql, 514
- client applications
 - using libpq library, 386
- client encoding
 - setting client encoding, 120
- client programs
 - connecting to PostgreSQL, 14
- client/server architecture, 14
 - client processing large amounts of data, 404
- CLOSE command, 556
- Close/close methods
 - NpgsqlConnection class, 522
 - NpgsqlDataReader class, 527
 - ResultSet interface, java.sql, 507
- closing database connections, PHP, 449
- CLUSTER command, 556
- Codd, E.F., 378
 - normalization, 33
 - RDBMS, 6, 8
- code independence
 - PEAR database abstraction interface, 460
- column aliases
 - SELECT statement, 81
- column constraints, 218
 - CHECK, 218
 - DEFAULT, 218
 - NOT NULL, 218
 - PRIMARY KEY, 218
 - REFERENCES, 218
 - UNIQUE, 218
- columns
 - see database columns
- command keywords, SQL
 - case sensitivity, 10
- command types, SQL, 9
- command-line applications
 - data access with PostgreSQL, 15
- command-line compilers
 - Npgsql in Mono, 523–524
- command-line versions
 - database management, 330
 - PostgreSQL configuration methods, 321
- command-lines
 - default command-line options, 311
 - ecpg arguments, 424
 - psql command line options, 118–119
 - table of, 573–574
 - vacuuming from, 351
- commands
 - execute shell command, 121
 - PostgreSQL commands, 551
 - syntax for SQL commands, 552–572
 - psql commands, 78, 118
 - command history, 115
 - internal commands, 119–121
 - issuing commands in psql, 114–115
 - reading from file, 115
 - showing commands sent to server, 330
 - tooggling timing of commands, 121
- CommandText/*~Timeout/**~Type* properties
 - NpgsqlCommand class, 526
- comma-separated values (CSV) file
 - using psql copy command, 159
- COMMENT command, 556
 - CREATE SCHEMA command, 332
- comments
 - block comments, 284
 - CREATE FUNCTION statement, 283
 - listing, 120
 - single-line comments, 284
 - stored procedures, 284
- commercial support, 13
- COMMIT command, 557
 - ecpg programs, 424
 - single user transactions, 248
 - transactions, 244
- commit method
 - Connection interface, java.sql, 499
- comparison operators (<,<=,>,>=), 272
 - choosing rows in SELECT, 87
 - dates and times, 99
- Comprehensive Perl Archive Network
 - see CPAN
- compression level, specifying, 340
- concatenation
 - string operators, 272
- CONCUR_READ_ONLY concurrency
 - type, 502
- CONCUR_UPDATEABLE concurrency
 - type, 502
- updateable result sets, 505

- concurrency types
 - JDBC result sets, 502
- conditional operators (AND/OR/NOT)
 - choosing rows in SELECT, 87, 88
- conditional statements
 - execution control structures, 291
- Conectiva Linux
 - packages for download available at, 44
- configuration
 - allowing remote connections, 313
 - authentication name mapping, 311
 - automatic postmaster startup, 57
 - client authentication options, 311
 - internal configuration, 320–338
 - main configuration file, 311
 - PostgreSQL.conf file options, 314
 - sample files, 316
 - system configuration, 309–316
 - utility to report PostgreSQL
 - configuration, 311
 - version information, 311
- configuration methods
 - PostgreSQL internal configuration, 320–321
- configure script
 - adding PostgreSQL support to PHP
 - installations, 446
 - installing PostgreSQL, 50
 - configure script options, 51
 - pg_config command, 52
- conformance
 - SQL levels of conformance, 8
- connect function, Perl DBI
 - connecting to PostgreSQL, 476
 - DBI environment variables, 475
 - Perl DBI features, 473
- connect method
 - Driver interface, java.sql, 498
- CONNECT statement
 - embedded SQL, 425
 - connection parameters, 426
- connecting to database
 - see* database connections
- connection attributes
 - Perl DBI database connections, 475
- connection handles
 - PHP making database connections, 447
 - connection parameters, 448
 - retrieving connection handle
 - information, 449
- Connection interface, java.sql
 - commit method, 499
 - createStatement method, 498, 499
 - creating database statements, 498
 - database connections, 498
 - getAutoCommit method, 499
 - getMetaData method, 500
 - getTransactionIsolation method, 500
 - handling database transactions, 499
 - prepareCall method, 499
 - prepareStatement method, 499
 - retrieving database metadata, 500
 - rollback method, 499
 - setAutoCommit method, 499
 - setTransactionIsolation method, 500
- connection parameters, 448
- connection pooling
 - Npgsql in Mono, 521
- Connection property
 - NpgsqlCommand class, 526
- connections to PostgreSQL
 - adding server connection in pgAdmin, 127
 - allowing remote connections, 313
 - allowing remote TCP/IP connections, 316
 - allowing secure database connections, 316
 - granting connection permissions, 54
 - logging connections to database, 315
 - logging disconnections from database, 315
 - max_connections, 314
 - PHP making database connections, 447
 - setting address for, 314
 - setting maximum number of, 314, 316
 - setting maximum number of
 - superusers, 314
 - setting port for, 314
 - specifying database name, 118
 - superuser_reserved_connections
 - option, 314
- ConnectionString property
 - NpgsqlConnection class, 522
- ConnStatusType
 - checking state of connection using
 - libpq, 389
- consistency
 - ACID transaction rules, 246
- CONSTANT modifier
 - variable declarations, 286
- constants
 - constant data, 2

- constraints
 - column constraints, 218
 - CREATE CONSTRAINT TRIGGER, 558
 - foreign key constraints, 232–242
 - primary key constraints, 219
 - SET CONSTRAINTS, 571
 - table constraints, 222
- continuation lines
 - prompt changes, 193
- continue action
 - whenever statement, EXEC SQL, 431
- contrib (PostgreSQL-contrib) binary
 - package, 44
- conversions
 - ALTER CONVERSION, 552
 - CREATE CONVERSION, 558
 - DROP CONVERSION, 563
 - listing, 120
- copy command (\copy), psql, 119,
 - 159–162, 574
 - loading data using, 161
 - NULL values, 160
 - sequence numbers, 161
 - syntax, 160
 - USING DELIMITERS option, 160
- COPY command, SQL, 160, 557
 - with data stream to client, 119
- copyright, PostgreSQL, 15
- copyright command (\copyright), psql,
 - 119, 574
- correlated subqueries, 188–191
 - execution of, 189
 - table aliases, 189
- cos function, 275
- cot function, 275
- count function, 174–182
 - count(*) function, 174–181
 - GROUP BY clause and, 176–178
 - HAVING clause and, 178–181
 - count(column name) function, 181–182
 - DISTINCT keyword, 182
 - updating data in database, 167
- COUNT statement
 - selecting data, 31
- CPAN (Comprehensive Perl Archive Network)
 - installing CPAN module, 466
 - installing DBI and DBD from source, 471
 - installing Perl modules, 466–467
- CREATE AGGREGATE command, 558
- CREATE CAST command, 558
- CREATE CONSTRAINT TRIGGER
 - command, 558
- CREATE CONVERSION command, 558
- CREATE DATABASE command, 558
 - database management, 329
- CREATE DOMAIN command, 559
- CREATE FUNCTION command, 559
 - add_one function, 279
 - comments, 283
 - defining functions, 276
 - PL/pgSQL function, 282
 - SQL functions, 298
 - using quotes in creation string, 281
- CREATE GROUP command, 559
 - PostgreSQL group configuration, 325
- CREATE INDEX command, 559
 - database performance, 352
- CREATE LANGUAGE command, 560
- CREATE OPERATOR CLASS command, 560
- CREATE OPERATOR command, 560
- CREATE RULE command, 560
- CREATE SCHEMA command, 560
 - schema management, 332
- CREATE SEQUENCE command, 561
- CREATE TABLE AS command, 562
- CREATE TABLE command, 217–218, 561
 - creating sample database tables, 67
 - foreign key constraint, 236–239
 - schema management, 333
 - order of table creation, 577
 - SQL introduction, 9
- CREATE TABLESPACE command, 327, 562
- CREATE TRIGGER command, 300, 562
- CREATE TYPE command, 562
- CREATE USER command, 562
 - PostgreSQL user configuration, 322
- CREATE VIEW command, 228–231, 563
- create_tables.sql file, 67
- createdb command, Linux/UNIX
 - creating sample database, 65
- CREATEDB option
 - CREATE USER command, 322
 - psql command-line tool, 75
- createdb utility
 - bin directory, 310
 - options, 330
- createdb.exe command, Windows
 - creating sample database, 66

- createlang utility, 277
 - bin directory, 311
 - installing procedural languages, 278
 - script options, 277
 - createStatement method
 - Connection interface, java.sql, 498, 499
 - createuser utility
 - bin directory, 310
 - creating user records, 65
 - options, 322
 - PostgreSQL user configuration, 322
 - creating database
 - see* database creation
 - crypt authentication methods
 - pg_hba.conf file, 312
 - CSV driver
 - DBD::CSV driver, 469
 - cube root operator, 271
 - currency
 - choosing data type, 376
 - CURRENT_XYZ (“magic”) variables
 - data manipulation, 215–216
 - curval function, psql
 - sequence numbers, 156, 161
 - cursors
 - CLOSE command, 556
 - DECLARE command, 563
 - FETCH command, 566
 - implementing in embedded SQL, 441–443
 - inserting data methods, ResultSet, 506
 - libpq using, 404–411
 - fetching all results at once, 406
 - fetching results in batches, 408
 - general structure of coding, 405
 - retrieving binary values, 411
 - MOVE command, 567
 - ResultSet interface querying cursor
 - position, 503
 - customer table
 - creating table, 67, 578
 - identifying primary keys, 372
 - populating sample database tables, 69
 - psql creating, 115
- D**
- d command (\d), psql, 119, 150, 574
 - examining database structure, 117
 - starting psql, 114
 - D option (-D)
 - createdb/dropdb utilities, 330
 - initdb utility, 54, 317
 - pg_ctl utility, 319
 - postmaster, 55
 - postmaster.opts file, 316
 - d option (-d)
 - createlang utility, 278
 - createuser utility, 323
 - database connections, 56
 - pg_dump utility, 341
 - pg_restore utility, 342
 - postmaster.opts file, 316
 - psql, 118, 573
 - vacuumdb utility, 351
 - da command (\da), psql, 174
 - da/db...du commands, 119, 574, 575
 - data
 - see also* transactions
 - Microsoft Excel importing, 143–145
 - phpPgAdmin tool importing, 132
 - programming with data, 1
 - retrieving data with ecpg, 436–440
 - data access, 23–28
 - accessing data across a network, 24
 - multiuser access, 25
 - ODBC .NET database connections, 519
 - postgres user, 53
 - PostgreSQL, 15
 - postmaster application, 53
 - projection (column selection), 27
 - pseudo users, 53
 - SELECT statement, 73–112
 - advanced features, 173–200
 - selection (row selection), 26
 - using Npgsql in Mono, 525–532
 - Npgsql event logging, 530
 - NpgsqlCommand class, 525
 - NpgsqlDataReader class, 527
 - retrieving metadata, 530
 - data columns
 - accessing data with projection, 27
 - choosing data types for, 21
 - database design, 33
 - spreadsheets, 18
 - storing data in databases, 21
 - Data Control Language (DCL)
 - SQL command types, 9

- Data Definition Language (DDL)
 - SQL command types, 9
- data directory
 - pg_hba.conf file, 311
 - pg_ident.conf file, 313
 - PostgreSQL installation, Linux/UNIX, 47
 - PostgreSQL.conf file, 313
 - postmaster.opts file, 315
 - secure PostgreSQL installations, 317
 - specifying location of, 317
 - system configuration, 311–316
 - user-accessible files in subdirectory, 311
- data entities
 - relationships between, 359
- data entry
 - Microsoft Access, 141
- data files
 - ownership of, 53
 - PostgreSQL files, 47
- data handling, 149–171
 - see also* deleting data from database;
inserting data into database;
transactions; updating data in
database
 - handling empty results, host variables, 439
 - logical unit of work, 244
 - support for large objects, 581–587
- data integrity
 - database design imposing, 359
 - RDBMS, 6
- data manipulation, 212–217
 - converting between data types, 212–214
 - CURRENT_XYZ (“magic”) variables,
215–216
 - functions for, 214–215
 - OID (object ID) column, 216–217
- Data Manipulation Language (DML)
 - SQL command types, 9
- data models
 - physical data model, 374
- data retrieval
 - see* data access
- data rows
 - accessing data with selection, 26
 - database design, 33
 - identifying rows uniquely, 22
 - primary keys, 22
 - spreadsheets, 18
- Data Source Names
 - see* DSNs
- data sources, PostgreSQL
 - configuring data source, 124
 - creating data source, 123, 124
- data storage
 - database management system, 4
 - databases, 21–23
 - adding information, 28–32
 - choosing data types for columns, 21
 - data types, 40
 - designing tables, 32–39
 - identifying columns required, 21
 - identifying rows uniquely, 22
 - NULLs, 41
 - relating tables using joins, 29
 - using multiple tables, 28
 - different file categories, 48
 - flat files, 2
 - file size problem, 4
 - repeating groups problem, 3
 - spreadsheets, 17–20
- data types, 201–212
 - ALTER TYPE, 555
 - arrays, 210–212
 - boolean data type, 202–204
 - changing data type in SELECT, 86
 - char data type, 40
 - character data types, 204–206
 - columns, choosing for, 21
 - converting between, 212–214
 - CREATE TYPE, 562
 - currency, choosing for, 376
 - data storage in databases, 40
 - date data type, 40, 94
 - DROP TYPE, 565
 - establishing during design, 375–376
 - geometric data types, 209–210
 - integer, 40
 - listing, 120
 - mapping Java to PostgreSQL, 505
 - money data type, 376
 - network data types, 209–210
 - number data types, 206–209
 - numeric data types, 40

- PostgreSQL data types, 545–549
 - approximate number, 546
 - character, 547
 - Classless Inter-Domain Routing, 548
 - date and time, 547
 - exact number, 546
 - geometric, 548
 - logical, 545
 - MAC address, 549
 - object identifier, 549
 - serial, 548
- psql command listing, 78
- serial, 40
- temporal data type, 209
- timestamp data type, 94
- varchar data type, 40
- data_sources function, Perl DBI, 483
- DataAdapter object
 - altering data, Npgsql in Mono, 537
 - ODBC .NET, 520
 - populating DataSet using, 538
 - relationships between ADO.NET objects, 537
- database abstraction interface, PEAR, 460
- database access
 - see* database connections
- database backup and recovery, 338–346
 - creating backup, 339–341
 - pgAdmin III, 343–346
 - restoring from backup, 341–343
 - utility to back up database, 311
- DATABASE column
 - pg_hba.conf file, 312
- database columns
 - average of values in column
 - see* avg function
 - changing data type in SELECT, 86
 - counting non NULL rows
 - see* count function
 - database limits, PostgreSQL, 544
 - maximum value in column
 - see* max function
 - minimum value in column
 - see* min function
 - OID (object ID) column, 216
 - selecting all columns, 79
 - selecting named columns, 80, 81
 - specifying when inserting data, 152–154
 - total sum of values in column
 - see* sum function
- database connections
 - accessing PostgreSQL from C#, 517–541
 - Npgsql in Mono, 520–539
 - Npgsql in Visual Studio, 539
 - ODBC .NET data provider on Windows, 517–520
 - accessing PostgreSQL from Java, 491–516
 - checking database functioning, 56
 - Connection interface, java.sql, 498
 - connection parameters, 448
 - ECPGstatus function, 427
 - embedded SQL, 425–427
 - connection parameters, 426
 - disconnecting, 427
 - example PHP script, 447
 - EXEC SQL specifying connection, 426
 - granting connection permissions, 54
 - JDBC API managing, 495
 - JDBC client connecting, 496
 - JDBC making database connections, 498–502
 - libpq, 387–391
 - creating new connection, 387–391
 - making asynchronous database connection, 415
 - log files, 47
 - Npgsql in Mono, 521–524
 - ODBC .NET data provider on Windows, 518
 - PEAR DB::connect function, 461
 - Perl DBI, 468, 473–477
 - PHP making, 447–450
 - connection information functions, 449
 - psql internal command, 119
 - Rekall, 134
 - sample database, 66
 - specifying database name, 118
- database creation
 - allowing user to create databases, 323
 - creating sample database, 64–72
 - creating tables, 67
 - creating database, 65
 - creating user records, 65
 - populating tables, 69
 - removing tables, 68
 - responsibilities of DBMS, 10
 - utilities to create/delete database, 310
- database design, 357–384
 - see also* database schema
 - additional resources, 384
 - business rules, 377

- checking design, 378
- creating simple database design, 34
 - completing initial design, 37
- data columns, 33
- data rows, 33
- data types, 375–376
- foreign keys, 373–375
- good database design, 357–360
 - accommodating future amendments, 360
 - data integrity, 359
 - holding required data, 358
 - identifying core tables, 359
 - performance, 360
 - planning with users, 357
 - satisfying requirements, 359
 - supporting data entity relationships, 359
- hierarchies, 381
- many-to-many relationships, 380–381
- naming conventions, 34
- normal forms, 378–380
- normalization, 33
- patterns in, 380–383
- physical model representation of, 371–378
- primary keys, 372–373
- recursive relationship pattern, 382–383
- repeating groups, 34
- stages in database design, 360–371
 - cardinality symbols, 367
 - converting entities to tables, 363
 - determining entities, 361
 - drawing relationship diagrams, 367
 - information gathering, 361
 - logical design development, 361
 - relating data entities, 366
 - validating conceptual design, 371
- table definitions, 377
- database directory
 - creating, 53
 - setting, 316
- database files
 - initdb locating, 54
 - installing on Windows, 59
 - postmaster locating, 55
- database front-end
 - Recall, 133
- database initialization, 317–318
 - default database installation, 317
 - installing PostgreSQL on Windows, 62
 - utility to initialize, 310
- Database Interface
 - see* Perl DBI
- database limits, PostgreSQL, 543–544
 - columns in a table, 544
 - database size, 543
 - field size, 544
 - row size, 544
 - rows in a table, 544
 - table indexes, 544
 - table size, 544
- database management
 - internal configuration, 328–331
 - using command-line, 330
- database management system, 4
- database models
 - full conceptual data model, 371
 - hierarchical model, 5
 - network model, 5
 - relational database model, 6
- Database option
 - Npgsql in Mono, 521
- database owner
 - setting user as new owner, 330
- database performance, 347–356
 - creating indexes, 352–356
 - monitoring operating system activity, 347
 - VACUUM command, 348–352
 - viewing PostgreSQL statistics, 348
 - utility to optimize database, 311
- Database property
 - NpgsqlConnection class, 522
- database rows
 - choosing rows in SELECT, 87–93
 - BETWEEN keyword, 89
 - IN keyword, 89
 - LIKE keyword, 91
 - LIMIT keyword, 92
 - operators restricting rows selected, 87, 89
 - counting rows
 - see* count function
- database limits, PostgreSQL, 544
- duplicated rows returned from SELECT, 83
 - suppressing duplicates, 84, 85
 - using DISTINCT keyword, 84
- grantable privileges, 337
- INSERT command, 567
- ORDER BY clause in SELECT, 81, 82
- printing rows only, 119

- SELECT command, 570
- showing, 121
- SQL functions returning multiple rows, 298
- UPDATE command, 572
- database schema
 - see also* database design
 - ALTER SCHEMA, 554
 - CREATE SCHEMA, 560
 - creating schemas, 332
 - creating tables in schemas, 333
 - designing database tables, 32
 - DROP SCHEMA, 565
 - dropping schemas, 333
 - final schema design, 577
 - listing, 120, 332
 - listing tables in schema, 336
 - order in which schema are searched, 315
 - public schema, 331
 - relating three or more tables, 106
 - schema management, 331–337
 - schemaname.tablename syntax, 336
 - setting schema search path, 334
- Database Server option
 - installing PostgreSQL on Windows, 60
- database servers
 - see also* servers
 - object layout inside, 328
 - server control, 318–320
 - server control, Linux and UNIX
 - running processes, 318
 - starting and stopping server, 319
 - specifying server host, 322, 330
 - waiting for server to come up, 319
 - not waiting, 319
- database tables
 - ALTER TABLE, 554
 - altering table structures, 223–227
 - column constraints, 218–221
 - converting entities to tables, 363
 - CREATE TABLE AS, 562
 - CREATE TABLE, 561
 - creating database design, 35
 - creating sample database tables, 67
 - psql creating, 115
 - creating tables, 217–218
 - creating tables in schemas, 333
 - database limits, 544
 - DELETE command, 563
 - deleting all data from, 170
 - deleting tables, 170, 227
 - describing, 119
 - designing, 32–39
 - DROP TABLE, 565
 - dropping sample database tables, 68
 - examining table properties with
 - pgAdmin, 128
 - listing, 119
 - LOCK command, 567
 - lookup tables, 377
 - multiple tables for data storage, 28
 - populating sample database tables, 69
 - psql command listing, 78
 - psql creating and populating, 115
 - psql describing, 150
 - relating tables
 - see* relating tables
 - schemaname.tablename syntax, 336
 - SELECT INTO command, 570
 - selecting from one table, 79
 - selecting from multiple tables, 100–110
 - selecting named columns, 80, 81
 - set table output option, 120
 - setting HTML table output mode, 118
 - setting HTML table tag options, 119
 - setting table output option, 120
 - setting table title for output, 119
 - table constraints, 222–223
 - table definitions, 377
 - table name aliases, 105
 - temporary tables, 227–228
 - TRUNCATE command, 571
 - turning on expanded table output, 119
- database views, 228–232
 - CREATE VIEW, 563
 - creating views, 228–231
 - DROP VIEW, 565
 - dropping views, 231
 - listing, 119
 - stored procedures/triggers, 306
- DatabaseMetaData interface, java.sql, 500
- databases
 - adding data to, 149–165
 - ALTER DATABASE, 553
 - CREATE DATABASE, 558
 - creating additional databases, 317
 - creating database, psql, 75
 - deleting data from, 169–171
 - deleting database, psql, 75

- deleting database, utility for, 310
- DROP DATABASE, 564
- examining database structure, 117
- initializing, 53
- listener process, 310
- listing, 120
- listing available databases, 118
- restoring database, utility for, 311
- specifying database location, 319
- specifying database server host, 118
- specifying database username, 119
- terminology, 21
- updating data in, 165–168
- VACUUM command, 572
- DataSet object
 - altering data, Npgsql in Mono, 537
 - ODBC .NET, 520
 - populating using DataAdapter, 538
 - relationships between ADO.NET objects, 537
- datasets
 - ODBC .NET database connections, 519
- DataTruncation class
 - JDBC API, 494
- date data type, 209, 547
 - component parts, 99
 - data types, 40, 94
- date_part function
 - using date and time functions, 99
- dates and times
 - changing default date handling, 96
 - comparison operators (<,=,>), 99
 - data types, 94
 - cast function, 96, 97, 98
 - date styles m/d/y or d/m/y, 94
 - default, 95
 - ISO, 8601 style, 94
 - setting datestyle, 95, 97
 - setting the style, 94
 - date_part function, 99
 - now function, 99
 - performing calculations with, 100
 - PGDATESTYLE environment variable, 95
 - SELECT statement, 94–100
 - using functions, 98–100
- datestyle option/variable
 - PGDATESTYLE environment variable, 95
 - PostgreSQL.conf file, 315
 - setting, 95, 97
 - SHOW command, psql, 97
- DB interface, PEAR, 460
- DBD (Database Driver) modules
 - DBD::ODBC, 469
 - DBD::PgPP, 469
 - Perl DBI, 468, 469
- DBD::CSV driver
 - Perl DBI, 469
- DBD::ODBC module, 469
 - installing, 470
 - installing DBI and DBD from source, 471
- DBD::Pg database driver
 - installing DBI and DBD from source, 471
 - module Pg, 465
- DBD::PgPP database driver/module, 469
 - connecting using, 474
 - connection options, 474
 - errors connecting, 476
 - installing, 470
 - installing DBI and DBD from source, 471
- DBI
 - see Perl DBI
- DBMS (database management system)
 - see RDBMS
- dbname connection option
 - PgPP driver, 474
 - PQconnectdb function, 388
- dbname connection parameter
 - pg_connect function, 448
- DbType property
 - NpgsqlParameter class, 533
- DB::connect function
 - error handling with PEAR, 461
- DB::isError function
 - error handling with PEAR, 461
- DCL (Data Control Language), 9
- DDL (Data Definition Language), 9
- deadlock_timeout option
 - PostgreSQL.conf file, 315
- deadlocks, 262–264
 - avoiding deadlocks, 264
 - example, 263
- DEALLOCATE command, 563
- Debian Linux
 - installing PostgreSQL, 44
- DEBUG exception level, 290
- debugging
 - ecpg preprocessor code, 443–444
 - logging embedded SQL execution, 425
 - setting level of debug information, 316

- declarations
 - ALIAS declaration, 285
 - PL/pgSQL functions, 284
 - RENAME declaration, 286
 - variable declarations, 286, 287
- declarative languages
 - programming languages, 28
- DECLARE command, 563
- DECLARE section
 - PL/pgSQL function declarations, 284
- DEFAULT keyword
 - column constraints, 218
 - host variables, embedded SQL, 432
 - inserting data into serial columns, 157
- default parameter values
 - RESET command, 568
- default_transaction_isolation option
 - PostgreSQL.conf file, 315
- default_with_oids option
 - PostgreSQL.conf file, 315
- deferrable constraints
 - foreign key constraint, 240
- DEFERRED keyword
 - foreign key constraint, 241
- degrees function, 274
- DELETE command, 563
- DELETE privilege
 - grantable privileges, 337
- DELETE statement, 169–170
 - see also* deleting data from database; TRUNCATE statement
 - deleting all rows from table, 171
 - executing SQL with libpq, 396
 - importance of WHERE clause, 169
 - ON DELETE keyword, 241–242
 - reporting rows affected, 429
 - triggers, 300
- deleteRow method
 - ResultSet interface, java.sql, 505
- deleting data from database, 169–171
 - see also* DELETE statement; TRUNCATE statement
 - deleting all data from table, 170
 - JDBC updateable result sets, 505
 - reasons for using stored
 - procedures/triggers, 306
 - using Npgsql in Mono, 536
- deleting functions, 281
- DELIMITERS option
 - copy command, psql, 160
- deny rules
 - pg_hba.conf file, 312
- deregisterDriver method
 - DriverManager class, java.sql, 494
- DESC keyword
 - ORDER BY clause in SELECT, 81, 82, 83
 - default sort order, 82
- designing databases
 - see* database design
- devel (PostgreSQL-devel) binary package, 45
- Development option
 - installing PostgreSQL on Windows, 60
- die function, Perl DBI
 - connecting to PostgreSQL, 476
- Direction property
 - NpgsqlParameter class, 533
- directories
 - base directory, 309, 310
 - changing working directory, 119
- dirty reads
 - PostgreSQL, 257
 - transaction isolation, 256–257
- disable option (-disable)
 - pg_dump utility, 341
- disconnect function, Perl DBI, 477
- DISCONNECT statement
 - EXEC SQL syntax, 427
- displaying queries, psql, 118
- Dispose method
 - NpgsqlCommand class, 526
 - NpgsqlDataReader class, 527
- DISTINCT keyword
 - avg function, 185
 - count(column name) function, 182
 - SELECT statement, 84, 85
 - disadvantages using, 85
 - relating three or more tables, 110
 - sum function, 184
- division operator, 271
- dl command (\dl), psql
 - importing and exporting images, 583
- dn command (\dn), psql
 - listing schemas, 332
- do action
 - whenever statement, EXEC SQL, 431
- do command (\do), psql, 78

- do function
 - executing SQL using Perl DBI, 477
- doc directory
 - PostgreSQL installation, 47
 - system configuration, 316
- docdir (with-docdir) option, 51
- docs (PostgreSQL-docs) binary package, 45
- Documentation option
 - installing PostgreSQL on Windows, 60
- dollar quoting
 - creating stored procedures, 282
 - variable declarations, 287
- domains
 - ALTER DOMAIN, 553
 - CREATE DOMAIN, 559
 - DROP DOMAIN, 564
 - listing, 120
- dosql function
 - executing SQL using Perl DBI, 477
 - XML_RDB module, Perl, 486
- doSQLquery function
 - binding parameters, Perl DBI, 482
 - statement handle attributes, Perl DBI, 480
- dot operator
 - string concatenation, 450
- double data type, 546
- Driver interface, java.sql
 - acceptsURL method, 498
 - connect method, 498
 - forName method, 496
 - getMajorVersion method, 498
 - getMinorVersion method, 498
 - implementing, 496
 - jdbcCompliant method, 498
- driver option (-driver)
 - sql2xml.pl script, 487
 - xml2sql.pl script, 488
- DriverManager class, java.sql
 - deregisterDriver method, 494
 - getConnection method, 495
 - getDriver method, 495
 - getDrivers method, 494
 - getLoginTimeout method, 496
 - JDBC URLs, 495
 - managing connections, 495
 - managing JDBC logging, 496
 - managing login timeouts, 496
 - methods for managing drivers, 494
 - primary function, 493
 - println method, 496
 - PrintWriter method, 496
 - registerDriver method, 494
 - setLoginTimeout method, 496
 - setLogWriter method, 496
 - specifying server and database, 495
- drivers
 - available_drivers function, Perl DBI, 483
- DROP AGGREGATE command, 563
- DROP CAST command, 563
- DROP CONVERSION command, 563
- DROP DATABASE command, 564
 - database management, 329
- DROP DOMAIN command, 564
- DROP FUNCTION command, 564
 - deleting functions, 281
- DROP GROUP command, 564
 - PostgreSQL group configuration, 326
- DROP INDEX command, 564
- DROP LANGUAGE command, 279, 564
- DROP OPERATOR CLASS command, 564
- DROP OPERATOR command, 564
- DROP RULE command, 564
- DROP SCHEMA command, 565
 - schema management, 333
- DROP SEQUENCE command, 565
- DROP TABLE command, 68, 170, 227, 565
- DROP TABLESPACE command, 328, 565
- DROP TRIGGER command, 300, 565
- DROP TYPE command, 565
- DROP USER command, 565
 - PostgreSQL user configuration, 324
- DROP VIEW command, 231, 565
- drop_tables.sql file, 68, 69
- dropdb options
 - command-line database management, 330
- dropdb utility, 310
- droplang utility, 311
- dropuser utility, 310
- DSNs (Data Source Names)
 - ODBC .NET database connections, 518
 - Perl DBI database connections, 474
- dT command (\dT), psql, 78
- dt command (\dt), psql, 76, 78
 - examining table structure, 117
- duplicate rows returned from SELECT
 - using DISTINCT keyword, 84
- durability
 - ACID transaction rules, 247

- dynamic queries, 297
 - creating general-purpose update, 297
 - FOR loops, 298
- E**
- e command (\e), psql, 120, 575
 - query history, 115
- E option (-E)
 - createdb/dropdb utilities, 330
 - psql, 118, 573
- e option (-e)
 - createdb/dropdb utilities, 330
 - createlang utility, 278
 - createuser utility, 323
 - pg_restore utility, 342
 - psql, 118, 573
- Easy module
 - DBIx::Easy, 484–485
- echo command (\echo), psql, 120, 575
- echoing input, psql, 118
- echoing queries, psql, 118
- EchoMessages property
 - NpgsqlEventLog, 531
- ecpg preprocessor
 - adding to directories searched, 424
 - arguments, 424
 - auto-commit mode, 424
 - automatically committing statements, 424
 - BEGIN ... COMMIT blocks, 424
 - C translated source code, 421
 - creating C file, 421
 - creating executable program, 422
 - debugging code, 443–444
 - embedded SQL, 419–424
 - calling libpq library, 419
 - debugging, 425
 - host variables, 432
 - makefile for ecpg programs, 423–424
 - reporting errors, 428–430
 - retrieving data with ecpg, 436–440
 - trapping errors, 431
 - using cursors with ecpg, 441–443
 - help, 424
 - include files, 421
 - naming output file, 424
 - not specifying -t option, 424
 - version and search path information, 424
 - writing esqlc program, 420–422
 - transactions, 420
- ECPGdebug function
 - logging embedded SQL execution, 425
- ECPGFLAGS variable
 - makefile for ecpg programs, 423
- ECPGstatus function
 - database connections, 427
- ECPGxyz functions
 - C translated source code, 421, 422
 - logging embedded SQL execution, 425
- editing
 - preventing line editing, 118
- embedded SQL
 - accessing PostgreSQL from C using, 419–444
 - compiler, 311
 - creating executable program, 422
 - data access with PostgreSQL, 15
 - debugging ecpg code, 443–444
 - ecpg preprocessor, 419–424
 - using cursors with, 441–443
 - ECPGxyz functions, 422
 - error handling, 427–431
 - esqlc program, writing, 420–422
 - host variables, 432–435
 - declaring fixed-length variable types, 432
 - retrieving data with ecpg, 436
 - variable-length data, 434, 435
 - implementing cursors, 441–443
 - keywords, case sensitivity, 421
 - logging SQL execution, 425
 - making database connections, 425–427
 - retrieving data with ecpg, 436–440
 - handling empty results, 439
 - NULL database values, 438
 - null-terminated strings, 437
 - transactions, 420
- empty results
 - retrieving data with ecpg, 439
- encoding
 - character encoding, PHP, 459
 - setting client encoding, 120
 - setting encoding for new database, 330
 - text string support for large objects, 582
- encoding command (\encoding), psql, 120, 575
- ENCODING option
 - CREATE DATABASE command, 329
- END command, 565
- enhancements
 - database design allowing, 360

- entities
 - converting entities to tables, 363
 - determining entities, 361
- entity relationship diagram
 - creating simple database design, 35
 - customers, orders and items, 370
 - full conceptual data model, 371
- environment variables
 - connecting to database using libpq, 388
- equal to operator, 87
- equality operator
 - operator precedence, 270
- error handling
 - DBI::Easy module, 485
 - embedded SQL, 427–431
 - describing error, 428
 - error codes, 429
 - reporting errors, 428–430
 - result codes, 428
 - trapping errors, 431
 - PHP and, 458–459
 - with PEAR, 461
 - error-handling behaviors, 462
 - trigger_error function, 462
- errors
 - setting logging level of detail, 314
 - use of DISTINCT keyword masking, 85
- escape characters
 - backslash (\), 151, 152
 - inserting a single quote, 152
 - internal commands, psql, 114
- esqlc program
 - writing, 420–422
- eval block
 - data_sources function, 483
 - errors connecting, 476
 - RaiseError attribute, 475
- events
 - asynchronous working, libpq, 412
 - event logging, Npgsql, 530
- exact number data types, 546
- Excel
 - see* Microsoft Excel
- exception classes
 - JDBC API, 494
- EXCEPTION exception level, 290
- exceptions
 - BatchUpdateException class, 494
 - execution control structures, 290
 - JDBC API SQL exceptions, 494
 - SQLException class, 494
- exclusive locks, 262
- EXEC SQL syntax
 - CONNECT statement, 425
 - DISCONNECT statement, 427
 - FETCH INTO statement, 441
 - making database connections, 425–427
 - SET statement, 427
 - specifying particular database connection, 426
 - whenever statement, 431
 - writing esqlc program, 421
- ExecStatusType enumeration
 - common values, 393
 - executing SQL with libpq, 392
- executable files
 - bin directory, 310
- EXECUTE command, 297, 566
- execute function
 - result sets, Perl DBI, 479
- execute method
 - PEAR, 462
 - PreparedStatement interface, java.sql, 513
 - Statement interface, java.sql, 509
- EXECUTE privilege
 - grantable privileges, 337
- executeBatch method
 - Statement interface, java.sql, 510
- ExecuteNonQuery method
 - NpgsqlCommand class, 526, 536
- executeQuery method
 - PreparedStatement interface, java.sql, 513
 - Statement interface, java.sql, 508
- ExecuteReader method
 - NpgsqlCommand class, 526, 527, 529
- executeUpdate method
 - PreparedStatement interface, java.sql, 513
 - Statement interface, java.sql, 509
- executing files
 - f command line option, psql, 116
 - sql extension to files, 116
- executing queries, psql, 118

- execution control structures
 - conditional statements, 291
 - CASE function, 292
 - IF-THEN-ELSE block, 291
 - NULLIF function, 292
 - PL/pgSQL, 289
 - exception levels, 290
 - exceptions and messages, 290
 - RAISE statement, 290
 - RETURN statement, 290
- execution path, adding resources to, 58
- execution plans
 - EXPLAIN command, 566
- existence subqueries, 191–192
- EXISTS keyword
 - existence subqueries, 191
- exp function, 274
- expanded table output
 - turning on, 119
- EXPLAIN command, 89, 566
 - ANALYZE option, 350
 - query plan, 349, 350
- explicit locking, 264–266
 - desirability of, 264
 - locking rows, 265–266
 - locking tables, 266
- explicit transactions, 261
- exponentiation operator, 271
 - operator precedence, 270
- F**
- f command (\f), psql, 120, 575
- F option (-F)
 - pg_dump utility, 340
 - pg_restore utility, 342
 - psql, 118, 573
- f option (-f)
 - pg_dump utility, 340
 - pg_restore utility, 342
 - psql, 118, 573
 - executing file, 116
 - vacuumdb utility, 351
- factorial operator, 271
- features, PostgreSQL, 11
- Fedora
 - packages for download, 44
- FETCH command, 566
- fetch direction and size
 - ResultSet interface manipulating, 504
- FETCH INTO statement
 - EXEC SQL syntax, 441
- FETCH statement
 - fetching results using cursors, libpq, 405
 - fetching all results at once, 406
 - fetching results in batches, 408
 - NEXT option, 411
 - fetchall_arrayref function, Perl DBI, 479
 - fetchrow_array function, Perl DBI, 479
 - fetchrow_arrayref function, Perl DBI, 479
 - fetchrow_hashref function
 - DBI::Easy module, 485
 - Perl DBI, 479
- field information
 - PHP functions and result sets, 456
- field separators
 - changing, 120
 - setting, 118
- FieldCount property
 - NpgsqlDataReader class, 527, 529
- fields
 - database limits, PostgreSQL, 544
- files
 - COPY command, 557
 - executing file, psql, 116
 - not reading startup file, 119
 - PostgreSQL, 47
 - data storage for different categories, 48
 - psql command executing commands
 - from, 78
 - sql extension to files, 116
 - structured text files, 3
- first method
 - ResultSet interface, java.sql, 503
- first normal form
 - database design, 378
- flat files
 - data storage using, 2
 - repeating groups problem, 3
 - structured text files compared, 3
- float data type, 207
- float functions, 274
- float8 data type, 207
- floor function, 274
- FOR loops
 - dynamic queries, 298
 - execution control structures, 293
 - REVERSE option, 294

- foreign keys
 - deleting all rows from table, 171
 - establishing in design, 373–375
 - foreign key constraint, 232–242
 - adding constraint to existing table, 233
 - ALTER TABLE command, 235
 - as column constraint, 233–234
 - as table constraint, 234–239
 - constraint options, 240–242
 - CREATE TABLE command, 236–239
 - deferrable constraints, 240
 - grantable privileges, 337
 - order of table creation, 577
 - forms
 - Recall creating forms, 135
 - forName method
 - Driver interface, java.sql, 496
 - Forward-only result sets, 502
 - FREEZE option
 - VACUUM command, 349
 - FROM clause
 - SELECT statement, 78
 - UPDATE statement, 168
 - front-end programs
 - using libpq library, 386
 - FULL option
 - VACUUM command, 349
 - function calls
 - data access with PostgreSQL, 15
 - functions
 - see also* stored procedures
 - aggregate functions
 - see* aggregate functions, SELECT
 - ALTER FUNCTION, 553
 - built-in functions, 273–275
 - CREATE AGGREGATE command, 558
 - CREATE CAST command, 558
 - CREATE FUNCTION, 559
 - creating functions
 - add_one function, 279
 - editing script files, 280
 - using quotes, 281
 - data manipulation, 214–215
 - DECLARE statement, 284
 - defining functions, 276
 - deleting functions, 281
 - DROP AGGREGATE command, 563
 - DROP FUNCTION, 564
 - function overloading, 279
 - listing, 120
 - listing functions, 281
 - PL/pgSQL
 - function arguments, 283
 - SQL functions, 298–299
 - string functions, 275
 - trigonometric functions, 274
 - using in SELECT statements, 273
 - variable declarations, 285
- ## G
- g command (\g), psql, 79, 120, 575
 - query history, 115
 - g option (-g), psql, 118
 - garbage collection
 - VACUUM command, 572
 - GBorg (gborg.PostgreSQL.org)
 - resources for PostgreSQL tools, 146
 - geometric data types, 548, 209–210
 - getAutoCommit method
 - Connection interface, java.sql, 499
 - GetBoolean method
 - NpgsqlDataReader class, 527
 - getBoolean method
 - ResultSet interface, java.sql, 504
 - getConcurrency method
 - ResultSet interface, java.sql, 502
 - getConnection method
 - DriverManager class, java.sql, 495
 - GetData function
 - XML_RDB module, Perl, 486
 - GetDateTime method
 - NpgsqlDataReader class, 527
 - GetDecimal method
 - NpgsqlDataReader class, 528
 - GetDouble method
 - NpgsqlDataReader class, 528
 - getDriver method
 - DriverManager class, java.sql, 495
 - getDrivers method
 - DriverManager class, java.sql, 494
 - getFetchDirection method
 - ResultSet interface, java.sql, 504
 - getFetchSize method
 - ResultSet interface, java.sql, 504
 - GetFieldType method
 - NpgsqlDataReader class, 528
 - retrieving metadata, 530

- GetFloat method
 - NpgsqlDataReader class, 528
 - getInt method
 - ResultSet interface, java.sql, 504
 - GetInt16/32/64 methods
 - NpgsqlDataReader class, 528
 - getLoginTimeout method
 - DriverManager class, java.sql, 496
 - getMajorVersion method
 - Driver interface, java.sql, 498
 - getMessage method
 - PEAR_Error object, 461
 - getMetaData method
 - Connection interface, java.sql, 500
 - ResultSet interface, java.sql, 507
 - getMinorVersion method
 - Driver interface, java.sql, 498
 - getMoreResults method
 - Statement interface, java.sql, 509
 - GetName method
 - NpgsqlDataReader class, 528
 - retrieving metadata, 530
 - getResultSet method
 - Statement interface, java.sql, 509
 - GetString method
 - NpgsqlDataReader class, 528
 - getString method
 - ResultSet interface, java.sql, 504
 - getTransactionIsolation method
 - Connection interface, java.sql, 500
 - getType method
 - ResultSet interface, java.sql, 502
 - getUpdateCount method
 - Statement interface, java.sql, 509
 - gmake command, GNU
 - installing PostgreSQL, 52
 - GNU tools
 - installing PostgreSQL, 50
 - using make command, 52
 - goto action
 - whenever statement, EXEC SQL, 431
 - GRANT command, 566
 - privilege management, 337
 - graphical tools
 - PostgreSQL configuration methods, 321
 - resources for PostgreSQL tools, 147
 - using Microsoft Excel, 142
 - greater than operator, 87
 - comparison operators, 272
 - GROUP BY clause, 176–178
 - SELECT statements, 174
 - using with HAVING clause, 179
 - group configuration
 - PostgreSQL internal configuration, 325–326
 - GROUP keyword
 - GRANT command, 337
 - groups
 - ALTER GROUP, 553
 - CREATE GROUP, 559
 - DROP GROUP, 564
 - listing, 120, 326
 - removing, 326
- H**
- H command (\H), psql, 575
 - internal commands, 120
 - h command (\h), psql, 78, 120, 575
 - supported SQL commands, 115
 - h option (-h)
 - createdb/dropdb utilities, 330
 - createlang utility, 278
 - createuser utility, 322
 - pg_dump utility, 341
 - pg_restore utility, 342
 - psql, 118, 573
 - specifying host, 56
 - starting psql, 114
 - vacuumdb utility, 351
 - H option (-H), psql, 118, 573
 - handlers
 - installing for procedural languages, 277
 - handling data
 - see* data handling
 - HasRows property
 - NpgsqlDataReader class, 527
 - HAVING clause, 178–181
 - SELECT statements, 174
 - WHERE clause compared, 178
 - header files
 - include directory, 316
 - using libpq library, 386
 - help
 - man directory, 316
 - psql command getting, 78, 120

- help option (-help)
 - createdb/dropdb utilities, 330
 - createuser utility, 323
 - ecpg arguments, 424
 - pg_dump utility, 340
 - pg_restore utility, 342
 - postmaster.opts file, 316
 - psql, 118
 - vacuumdb utility, 351
- hierarchies
 - database design, 381
 - database models, 5
- history
 - command history in psql, 115
 - printing, 120
- host based authentication (hba) file
 - see* pg_hba.conf file
- host connection option
 - PgPP driver, 474
 - PQconnectdb function, 388
- host connection parameter
 - pg_connect function, 448
- host variables
 - embedded SQL, 432–435
 - declaration sections, 432
 - declaring fixed-length variable types, 432
 - using DEFAULT keyword, 432
 - variable-length data, 434
 - handling empty results, 439
 - implementing cursors in embedded SQL, 441
 - indicator variable, 438
 - naming conventions, 438
 - NULL database values, 438
 - null-terminated strings, 437
 - retrieving data with ecpg, 436
- hostaddr connection option/parameter
 - pg_connect function, 448
 - PQconnectdb function, 388
- hosts
 - specifying database server host, 118
- HTML
 - setting HTML table output mode, 118
 - setting HTML table tag options, 119, 121
 - toggleing HTML mode, 120
- - i command (\i), psql, 78, 120, 575
 - creating sample database tables, 67
 - executing files, 116
 - reading psql commands from file, 115
 - I option (-I)
 - ecpg arguments, 421, 424
 - ecpg include file location, 422
 - using libpq library, 386
 - i option (-i)
 - createuser utility, 323
 - postmaster, 55
 - postmaster.opts file, 316
 - rpm install, 45
 - ident authentication methods
 - pg_hba.conf file, 312
 - IF-THEN-ELSE block
 - execution control structures, 291
 - images
 - importing and exporting using BLOBs, 583–585
 - programming BLOBs, 586–587
 - remote importing and exporting using BLOBs, 585–586
 - support for large objects, 581–587
 - using BLOBs, 583
 - using encoded text strings, 582
 - using links, 581–582
 - implementation classes
 - JDBC API, 492
 - Driver interface, 496
 - implicit transactions, 261
 - IN GROUP option
 - CREATE USER command, 322
 - IN keyword
 - choosing rows in SELECT, 89
 - used in subquery, 187
 - IN operator
 - operator precedence, 270
 - include directive
 - SQL control area (sqlca), 430
 - include directory
 - PostgreSQL installation, 47
 - system configuration, 316
 - include files
 - ecpg preprocessor, 421

- includedir option
 - pg_config command, 52
- indexes
 - ALTER INDEX, 553
 - CLUSTER command, 556
 - CREATE INDEX, 559
 - database limits, PostgreSQL, 544
 - database performance, 352–356
 - DROP INDEX, 564
 - PHP functions and result sets, 453–455, 457
 - REINDEX command, 568
 - using index on large table, 353
- indicator variables, 438
- indices
 - listing, 119
- inequality operator
 - operator precedence, 270
- inet data type, 209, 549
- INFO exception level, 290
- information gathering
 - database design stages, 361
- Ingres
 - PostgreSQL history, 12
- INHERITS keyword
 - CREATE TABLE command, 217
- initdb utility/file
 - bin directory, 310
 - initializing database, 54
 - location of database files, 54
 - options, table of, 317
 - postgres user running, 54
 - PostgreSQL installations, 317
- INITIALLY DEFERRED keywords
 - foreign key constraint, 240
- input option (-input)
 - xml2sql.pl script, 488
- INSERT command, 567
 - adding to DataAdapter object, 538
- insert function
 - DBI::Easy module, 485
- INSERT privilege
 - grantable privileges, 337
- INSERT statement, 149–159
 - see also* inserting data into database
 - inserting data into serial columns, 154–157
 - DEFAULT keyword, 157
 - sequence numbers, 155–157
 - inserting NULLs into database, 151, 158–159
 - INTO option, 153
 - loading data from another application, 162–165
 - OID (Object IDentification) number, 151
 - reporting rows affected, 429
 - transaction example, 250, 251
 - triggers, 300, 302
 - using safer syntax, 152–154
 - using simple syntax, 149–152
 - caution when using, 150
- inserting data into database
 - see also* INSERT statement
 - character data, 151
 - copy command, psql, 159–162
 - COPY command, SQL, 160
 - identifying number of rows inserted, 164
 - inserting NULL values, 158–159
 - loading data from another application, 162–165
 - JDBC updateable result sets, 506
 - using Npgsql in Mono, 536
- insertRow method
 - ResultSet interface, java.sql, 507
- installations, Perl
 - Perl DBI, 468–472
 - Perl modules, 466–468
- installations, PostgreSQL
 - see also* PostgreSQL installation on
 - Linux/UNIX; PostgreSQL installation on Windows
 - JDBC driver, 493
 - ODBC drivers, 121
 - pgAdmin III tool, 125
 - phpPgAdmin tool, 130
- int4 functions, 274
- integer data type, 40, 207, 546
- internal commands, psql
 - listing all, 119
 - psql command types, 114
 - psql command-line tool, 119–121
 - table of, 574–576
- internal configuration
 - see under* PostgreSQL
- internal variable
 - setting, 121
 - unsetting/deleting, 121
- interval data type, 209, 547

- INTO clause
 - INSERT statement, 153
 - retrieving data with `ecpg`, 436
- `ipcclean` utility, 311
- IS keyword
 - NULLs, 93
- IS operator
 - operator precedence, 270
- `isAfterLast` method
 - ResultSet interface, `java.sql`, 503
- `isBeforeFirst` method
 - ResultSet interface, `java.sql`, 503
- `IsClosed` property
 - `NpgsqlDataReader` class, 527
- `IsDBNull` method
 - `NpgsqlDataReader` class, 528
- `isFirst` method
 - ResultSet interface, `java.sql`, 503
- `isLast` method
 - ResultSet interface, `java.sql`, 503
- ISNULL operator
 - operator precedence, 270
- IsNullability property
 - `NpgsqlParameter` class, 533
- ISO, 8601 date style, 94
- isolation
 - ACID transaction rules, 246
 - handling multi-user access, 26
 - transaction isolation levels
 - changing, 261
 - PostgreSQL default, 261
 - undesirable phenomena, 256
 - transactions, 255–261
 - ANSI isolation levels, 260–261
 - changing isolation level, 261
 - dirty reads, 256–257
 - lost updates, 258–260
 - performance, 255
 - phantom reads, 258
 - unrepeatable reads, 257
- Item property
 - `NpgsqlDataReader` class, 527, 528
- item table
 - creating table, 67, 578
 - identifying primary keys, 373
 - populating sample database tables, 70
 - `psql` creating, 116
 - schema for item and stock tables, 196
- J**
- Java
 - accessing PostgreSQL from, 491–516
 - java class, `PostgreSQLMetaData`, 500
 - Java Database Connectivity (JDBC)
 - accessing PostgreSQL from Java, 491–516
 - API described, 491
 - implementation classes, 492
 - connecting to PostgreSQL, 14
 - core API, 491
 - data access with PostgreSQL, 15
 - extension API, 491
 - installing PostgreSQL JDBC driver, 493
 - JDBC clients using statements, 498
 - JDBC result sets, 502–507
 - JDBC statements, 507–516
 - making database connections, 498–502
 - mapping Java data types to PostgreSQL, 505
 - using PostgreSQL JDBC driver, 491–498
 - Driver interface, 496
 - DriverManager class, 493
 - `java.sql.CallableStatement` interface, 507
 - `java.sql.Connection` interface
 - see* Connection interface, `java.sql`
 - `java.sql.Driver`
 - see* Driver interface, `java.sql`
 - `java.sql.DriverManager`
 - see* DriverManager class, `java.sql`
 - `java.sql.PreparedStatement` interface
 - see* PreparedStatement interface, `java.sql`
 - `java.sql.ResultSet` interface
 - see* ResultSet interface, `java.sql`
 - `java.sql.Statement` interface
 - see* Statement interface, `java.sql`
 - `java.sql.Types` class, 505
 - `jdbc` (PostgreSQL-jdbc) binary package, 45
 - JDBC API
 - creating database statements, 498
 - creating JDBC statements, 507
 - definition and implementation layers, 492
 - exception classes, 494
 - future development, 516
 - handling database transactions, 499
 - implementing Driver interface, 496
 - managing connections, 495
 - managing JDBC logging, 496
 - managing login timeouts, 496
 - methods for managing drivers, 494
 - retrieving database metadata, 500
 - SQL exceptions and warnings, 494

- jdbc directory, 310
- JDBC driver option
 - installing PostgreSQL on Windows, 60
- JDBC drivers
 - classifications, 492
 - type 1/type 2/type 3/type 4, 492, 493
 - using, 491–498
- JDBC statements, 507–516
 - executing statements, 508–509
 - handling SQL batches, 509–510
 - querying results and result sets, 509
 - using prepared statements, 512–516
 - using statements, 508–512
- JDBC URLs, 495
- jdbcCompliant method
 - Driver interface, java.sql, 498
- JOIN ... ON clause
 - relating tables, 111
- join function, Perl DBI, 479
- joining tables
 - see* relating tables
- joins
 - outer joins, 196–200
 - relating tables using joins, 29
 - self joins, 194–196
 - UNION join, 192–194

K

- keys
 - see also* foreign keys; primary keys
 - network database models, 5
 - surrogate keys, 23
- keywords
 - case sensitivity, embedded SQL, 421
 - PL/pgSQL, 285
- KPackage
 - PostgreSQL installation, Linux, 48, 49

L

- l command (\l), psql, 120, 575
- L option (-L)
 - createlang utility, 278
 - creating executable program, 422
 - libpq library installation, 386
- l option (-l)
 - createlang utility, 278
 - pg_ctl utility, 319
 - pg_restore utility, 342
 - postmaster.opts file, 316

- psql, 118, 573
- rpm list, 48
- LANGUAGE clause
 - specifying procedural language, 276
- language extensions
 - utility to add/delete support for, 311
- languages
 - ALTER LANGUAGE, 554
 - block-structured languages, 282
 - CREATE LANGUAGE, 560
 - DROP LANGUAGE, 564
 - procedural languages, 276–282
 - programming languages, 28
- large objects
 - deleting, 584
 - listing, 119, 120
 - performing large object operations, 120
 - PostgreSQL support for, 581–587
 - using BLOBs, 583–587
 - using encoded text strings, 582
 - using links, 581–582
- last method
 - ResultSet interface, java.sql, 503
- LD_LIBRARY_PATH
 - ecpg shared library files, 422
- LDLIBS variable
 - makefile for ecpg programs, 423
- LEFT OUTER JOIN keywords, 198, 200
- length function, 214
 - character data types, 206
- less than operator, 87
 - comparison operators, 272
- Level property
 - NpgsqlEventLog, 531
- lib directory
 - PostgreSQL installation, 47
 - system configuration, 316
- libdir option
 - pg_config command, 52
- libpq functions
 - creating new function, 394
 - PQclear, 392
 - PQcmdTuples, 397
 - PQconnectdb, 387
 - PQconnectPoll, 415
 - PQconnectStart, 415
 - PQconsumeInput, 414
 - PQerrorMessage, 391
 - PQexec, 392, 394

- PQfinish, 389, 391
- PQflush, 414
- PQfname, 397
- PQfnumber, 398
- PQfsize, 398
- PQgetisnull, 401
- PQgetlength, 399
- PQgetResult, 412
- PQgetvalue, 399
- PQisBusy, 414
- PQisnonblocking, 413
- PQnfields, 397
- PQntuples, 397
- PQprint, 401
- PQrequestCancel, 415
- PQreset, 389
- PQresStatus, 394
- PQresultErrorMessage, 394
- PQresultStatus, 392
- PQsendQuery, 412
- PQsetnonblocking, 412
- PQstatus, 391
- libpq library
 - accessing PostgreSQL from C using, 385–417
 - asynchronous working, 411–417
 - canceling queries, 415
 - executing queries, 412
 - making asynchronous database connection, 415
 - nonblocking mode, 411
 - creating executable program, 422
 - database connections using, 387–391
 - checking state of connection, 389
 - closing connection, 389
 - connecting using environment variables, 388
 - connection parameters, 391
 - creating new connection, 387–391
 - resetting connection, 389
 - retrieving information about
 - connection errors, 391
 - using a makefile, 390–391
 - writing connection program, 389
 - drawback to using, 419
 - executing SQL with libpq, 392–401
 - binary values, 411
 - determining query status, 392–394
 - executing queries with PQexec, 394–396
 - extracting data from query results, 397–400
 - handling NULL results, 400
 - updating or deleting rows, 396
 - user specified data in query, 396
 - using cursors, 404–411
 - importing and exporting images, 586
 - installation, 386
 - managing transactions, 404
 - printing query results, 401–403
 - program compilation, 387
 - program structure, 386
 - using, 386–387
- library files
 - lib directory, 316
 - LOAD command, 567
- libs (PostgreSQL-libs) binary package, 44
- LIKE keyword
 - choosing rows in SELECT, 91
- LIKE operator
 - operator precedence, 270
 - string operators, 272
- LIMIT keyword
 - choosing rows in SELECT, 92
 - OFFSET clause, 92
- limits
 - see* database limits, PostgreSQL
- line data type, 209, 548
- line editing
 - preventing line editing, 118
- linking tables
 - Microsoft Access, 137
 - relating tables, 101
- links
 - support for large objects, 581–582
- Linux
 - installing PostgreSQL on, 43–58
 - from Linux binaries, 44
 - running processes on, 318
 - starting and stopping server on, 319
 - starting up psql, 74
- LISTEN command, 567
- listen_addresses option
 - PostgreSQL.conf file, 314
- listener process
 - databases, 310
 - UNLISTEN command, 572

- listing
 - database objects, 119–120
 - databases, 120
 - functions, 281
- ln function, 274
- lo_export (\lo_export) command, psql, 120, 575, 586
- lo_export function, 584, 586
- lo_import (\lo_import) command, psql, 120, 575, 585
- lo_import function, 583, 584, 586
- lo_list (\lo_list) command, psql, 120, 575, 583, 585
- lo_unlink (\lo_unlink) command, psql, 120, 575, 586
- lo_unlink function, 584, 586
- LOAD command, 567
- lobjects
 - see* large objects
- LOCK command, 567
- locking, 262–266
 - deadlocks, 262–264
 - exclusive locks, 262
 - explicit locking, 264–266
 - locking rows, 265–266
 - locking tables, 266
 - shared locks, 262
- LOG exception level, 290
- log files
 - appending server log messages, 319
 - PostgreSQL files, 47
 - transactions, 247
- log functions, 274
- log_connections option
 - PostgreSQL.conf file, 315
- log_destination option
 - PostgreSQL.conf file, 314
- log_disconnections option
 - PostgreSQL.conf file, 315
- log_error_verbosity option
 - PostgreSQL.conf file, 314
- log_min_messages option
 - PostgreSQL.conf file, 314
- logarithm operator, 270
- logging
 - event logging, Npgsql, 530
 - log tracing in progress, 532
 - managing JDBC logging, 496
 - logging embedded SQL execution
 - ECPGdebug function, 425
 - logging parameter, 425
 - logical data types, 545
 - logical design development
 - database design stages, 361
 - logical unit of work
 - transactions, 244
 - Logname property
 - NpgsqlEventLog, 531
 - lookup tables, 377
 - loops
 - execution control structures, 292
 - BEGIN ... END blocks, 293
 - FOR loops, 293
 - indefinite loop, 293
 - WHILE loops, 293
 - lost updates
 - transaction isolation, 258–260
 - lower function, 275
 - lseg data type, 209, 548
- M**
- m option (-m)
 - pg_ctl utility, 319
- macaddr data type, 209, 549
- magic variables
 - data manipulation, 215–216
- make clean command
 - makefile for ecpg programs, 423
- make command
 - compiling programs using makefile, 391
- make command, GNU
 - installing PostgreSQL, 52
- makefile
 - compiling programs, 390–391
 - ecpg programs, 423–424
- makemap function
 - DBI::Easy module, 485
- man directory
 - PostgreSQL installation, 47
 - system configuration, 316
- Mandrake
 - packages for download available at, 44
- MANPATH
 - adding resources to execution path, 58
 - viewing manual pages, 316
- manual pages
 - adding to execution path, 58

- many-to-many relationships
 - database design, 380–381
- mathematical functions, 274
- max function, 183–184
 - description, 174
 - NULL values, 184
 - sequence numbers, 156
 - varchar type columns, 183
- max_connections
 - PostgreSQL.conf file, 314
- maximum value in column
 - see* max function
- MaxPoolSize option
 - Npgsql in Mono, 521
- md5 authentication methods
 - pg_hba.conf file, 312, 313
- median function, 185
- memory
 - setting working memory, 314
- messages
 - appending server log messages, 319
 - execution control structures, 290
 - logging level for messages, 314
 - logging server messages, 314
 - running quietly, 118, 330
- metadata
 - getMetaData method, ResultSet, 507
 - JDBC retrieving database metadata, 500
- METHOD column
 - pg_hba.conf file, 312
- Microsoft Access
 - adding link table, 138
 - browsing link table, 140
 - creating blank Access database, 138
 - creating reports, 141
 - data entry, 141
 - selecting ODBC data source, 139
 - selecting tables to link, 139
 - using with PostgreSQL, 137–142
 - reasons for using Access, 137
 - using linked tables, 137
- Microsoft Excel
 - choosing an import location, 145
 - choosing columns to import into, 143
 - defining sort criteria for imported data, 144
 - importing data into, 143
 - restricting rows to import, 144
 - using with PostgreSQL, 142–146
 - charting, 142
 - Excel chart using PostgreSQL data, 146
 - graphical tools, 142
 - viewing imported data in, 145
- min function, 182–183
 - description, 174
 - NULL values, 183
 - varchar type columns, 183
- minimum value in column
 - see* min function
- MinPoolSize option
 - Npgsql in Mono, 521
- mod function, 274
- mode function, 185
- models
 - database models, 4
- modes
 - setting single-line mode, 119
 - setting single-step mode, 118
 - shutdown mode, 319
 - silent mode, 319
- module Pg
 - DBD::Pg database driver, 465
- modules
 - bundles, 469
- modulo operator, 271
- money data type, 207, 376, 546
- Mono
 - see* Npgsql in Mono
- mono command
 - compiling with command-line compiler, 524
- MonoDevelop
 - C# code retrieving data in, 529
 - compiling with, Npgsql in Mono, 524
- MOVE command, 567
- moveToCurrentRow method
 - ResultSet interface, java.sql, 506
- moveToInsertRow method
 - ResultSet interface, java.sql, 506
- multiple tables
 - selecting from, 100–110
- multiplication operator, 271
 - operator precedence, 270
- multiplying SQL results
 - performing calculations in SELECT, 86

- multitasking
 - responsibilities of DBMS, 11
- multiuser access
 - transactions, 244, 246
 - double booking error, 244
 - using databases for data access, 25
- N**
- N option (-N)
 - postmaster.opts file, 316
- n option (-n), psql, 118, 573
- name
 - reserved keywords, 362
- Name attribute, Perl DBI, 475
 - statement handles, 480
- names
 - choosing data type, 376
 - database design, 363
 - column aliases, 81
- naming conventions
 - database design, 34
 - embedded SQL keywords, 421
 - indicator variables, 438
 - sequence numbers, 155
 - variable names, 285
- Natural language support option
 - installing PostgreSQL on Windows, 60
- nesting transactions, 254
- .NET Framework
 - open source implementation of, 520
- network data types, 209–210
- network model
 - database models, 5
- networks
 - accessing data across a network, 24
 - granting PostgreSQL connection
 - permissions, 55
- NEW trigger procedure variable, 303
- next method
 - ResultSet interface, java.sql, 503
- NEXT option, FETCH statement
 - fetching results using cursors, libpq, 411
- nextval function
 - sequence numbers, 155, 156
- nonblocking mode
 - asynchronous working, libpq, 411
- normalization
 - database design, 33, 378–380
 - first normal form, 378
 - second normal form, 379
 - third normal form, 379
- not equal to operator, 87
- not found condition
 - whenever statement, EXEC SQL, 431
- NOT LIKE operator
 - string operators, 272
- NOT NULL clause
 - column constraints, 218
 - variable declarations, 286
- NOT operator
 - Binary NOT operator, 271
 - operator precedence, 270
- NOTICE exception level, 290
- notifications
 - LISTEN command, 567
 - NOTIFY command, 567
 - UNLISTEN command, 572
- NOTIFY command, 567
- NOTNULL operator
 - operator precedence, 270
- now function
 - using date and time functions, 99
- Npgsql driver option
 - installing PostgreSQL on Windows, 60
- Npgsql in Mono
 - accessing PostgreSQL from C#, 520–539
 - changing data
 - ExecuteNonQuery method, 536
 - using DataAdapter object, 537
 - using NpgsqlCommand object, 536
 - compiling with command-line compiler, 523–524
 - compiling with MonoDevelop, 524
 - connecting to database, 521–524
 - creating NpgsqlConnection object, 521–523
 - event logging, 530
 - NpgsqlEventLog properties, 531
 - retrieving data from database, 525–532
 - NpgsqlCommand class, 525
 - NpgsqlDataReader class, 527
 - retrieving metadata, 530
 - using parameters, 532
 - using prepared statements, 535
 - using statements, 533
- Npgsql in Visual Studio
 - accessing PostgreSQL from C#, 539

- NpgsqlCommand class
 - methods, 526
 - properties, 526
 - retrieving data from database, 525
 - NpgsqlCommand object
 - ExecuteNonQuery method, 536
 - NpgsqlConnection class
 - connecting to database, 521–523
 - connection string options, 521
 - methods, 522
 - properties, 522
 - NpgsqlDataReader class
 - methods, 527
 - properties, 527
 - retrieving data from database, 527
 - NpgsqlDbType enumeration, 533
 - NpgsqlDbType property
 - NpgsqlParameter class, 533
 - NpgsqlEventLog
 - properties, 531
 - NpgsqlParameter class
 - properties, 533
 - using Npgsql in Mono, 532
 - creating prepared statements, 535
 - creating statements with parameters, 533
 - NULL database values
 - retrieving data with `ecpg`, 438
 - NULL pointers
 - PQfinish/PQstatus functions, `libpq`, 391
 - NULLABLE attribute
 - statement handles, Perl DBI, 480
 - NULLIF function
 - execution control structures, 292
 - NULLs
 - aggregate functions, `SELECT`, 183
 - choosing data type, 375
 - comparing NULL values, 41
 - copy command, `psql`, 160
 - data storage in databases, 41
 - inserting into database, 151, 158–159, 435
 - IS keyword, 93
 - `libpq` handling NULL results, 400
 - logic behind use of, 93
 - outer joins, 197, 198
 - PHP definition of, 456
 - `SELECT` statement
 - checking for, 93–94
 - UNIQUE option, 219
 - null-terminated strings
 - retrieving data with `ecpg`, 437
 - NUM_OF_FIELDS attributes
 - statement handles, Perl DBI, 480
 - NUM_OF_PARAMS attributes
 - statement handles, Perl DBI, 480
 - number data types, 206–209
 - approximate number, 546
 - exact number, 546
 - numeric data type, 207, 546
 - data types, 40
- 0**
- o command (`\o`), `psql`, 120, 575
 - redirecting query output, 117
 - O option (`-O`)
 - `createdb/dropdb` utilities, 330
 - o option (`-o`)
 - `ecpg` arguments, 424
 - `pg_ctl` utility, 319
 - `psql`, 118, 573
 - object identifier (oid) data type, 549
 - object operator
 - operator precedence, 270
 - objects
 - `COMMENT` command, 556
 - `octet_length` function, 275
 - ODBC (Open Database Connectivity)
 - accessing data across a network, 24
 - accessing PostgreSQL from C#, 517–520
 - connecting to PostgreSQL, 14
 - creating data source, 123
 - data access with PostgreSQL, 15
 - Microsoft Access selecting data source, 139
 - ODBC Data Sources applet, 122
 - ODBC drivers, 121
 - installed ODBC drivers, 122, 123
 - installing, 121
 - standard Windows installation, 121
 - setting up, 121–125
 - `odbc` (PostgreSQL-`odbc`) binary package, 45
 - ODBC .NET Data Provider on Windows
 - accessing PostgreSQL from C#, 517–520
 - connecting to database, 518
 - retrieving data into dataset, 519
 - setting up ODBC provider, 517
 - `odbc` directory, 310

- ODBC driver
 - see also* DBD::ODBC database driver
 - installing PostgreSQL on Windows, 60
 - OFFSET clause
 - choosing rows in SELECT
 - using alone in SELECT, 92
 - using with LIMIT keyword, 92
 - OID (object ID) column
 - BLOBs, 583
 - data manipulation, 216–217
 - OID (Object Identification) number, 151
 - creating tables with OIDs, 315
 - data type, 549
 - OLD trigger procedure variable, 303
 - OLEDB provider option
 - installing PostgreSQL on Windows, 60
 - ON clause
 - outer joins, 198, 199
 - ON DELETE keyword
 - foreign key constraint, 241–242
 - ON UPDATE keyword
 - foreign key constraint, 241–242
 - online documentation
 - doc directory, 316
 - opaque return type, 300
 - Open Database Connectivity
 - see* ODBC
 - Open method
 - NpgsqlConnection class, 522
 - open-source software, 15
 - copyright, 15
 - main feature, 13
 - operators, 268–273
 - additional operators, 273
 - ALTER OPERATOR CLASS, 554
 - ALTER OPERATOR, 554
 - arithmetic operators, 270–271
 - precedence, 269
 - associativity, 269
 - built-in function equivalents, 274
 - comparison operators, 272
 - CREATE OPERATOR CLASS, 560
 - CREATE OPERATOR, 560
 - DROP OPERATOR CLASS, 564
 - DROP OPERATOR, 564
 - greater than (>) operator, 268
 - listing, 120
 - multiplication (*) operator, 268
 - order of precedence, 269, 270
 - performing regular expression matches, 268
 - psql command listing, 78
 - string operators, 272
 - options
 - psql command line options, 118–119
 - setting options to pass to postmaster, 319
 - OR operator
 - Binary OR operator, 271
 - conditional operator, 89
 - operator precedence, 270
 - ORDER BY clause
 - SELECT statement, 81, 82
 - ASC keyword, 81
 - default sort order, 82
 - DESC keyword, 81
 - example using ASC and DESC, 82, 83
 - sorting on unselected columns, 83
 - orderinfo table
 - creating table, 68, 578
 - identifying primary keys, 373
 - orderline table
 - creating table, 68, 579
 - identifying primary keys, 373
 - populating sample database tables, 71
 - outer joins, 196–200
 - NULL values, 197, 198
 - Oracle and DB2, 198
 - SQL92 standard, 198
 - output
 - running quietly, 118
 - toggling expanded output, 121
 - output modes
 - setting HTML table output mode, 118
 - output option (-output)
 - sql2xml.pl script, 487
 - overloading
 - function overloading, 279
 - owner
 - database owner, 330
- P**
- p command (\p), psql, 120, 575
 - query history, 115
 - P option (-P)
 - createuser utility, 323
 - creating user records, 65
 - psql, 118, 574

- p option (-p)
 - createdb/dropdb utilities, 330
 - createlang utility, 278
 - createuser utility, 322
 - pg_dump utility, 341
 - pg_restore utility, 342
 - psql, 114, 118, 573
 - vacuumdb utility, 351
- p port option (-p port)
 - postmaster.opts file, 316
- packages
 - see also* RPM (RPM Package Manager)
 - binary packages, 44
 - examining contents with KPackage, 49
 - selecting with YaST2, 45
- ParameterName property
 - NpgsqlParameter class, 533
- parameters
 - see also* arguments
 - binding to column names, 538
 - RESET command, 568
 - SET command, 570
 - setting data type, 538
 - SHOW command, 571
 - SQL functions, 298
 - using with Npgsql in Mono, 532–533
- Parameters property
 - NpgsqlCommand class, 526
- password authentication methods
 - pg_hba.conf file, 312
- password connection parameter/option
 - pg_connect function, 448
 - PQconnectdb function, 388
- Password option
 - Npgsql in Mono, 521
- passwords
 - prompting for, 330
- passwords, psql
 - password required, 117
 - prompting for password, 119
- PATH
 - adding resources to execution path, 58
- path data type, 548
- paths
 - setting schema search path, 334
- pattern matching
 - choosing rows in SELECT, 91
- PEAR (PHP Extension and Application Repository)
 - error handling with, 461
 - preparing and executing queries, 462
 - using, 459–463
 - using database abstraction interface, 460
 - wildcard characters, 462
- PEAR_Error object, 461
 - class information, 462
 - getMessage method, 461
 - setErrorHandler method, 462
- PEAR_ERROR_XYZ
 - error-handling behaviors, 462
- PERFORM statement
 - assignments, 289
- performance
 - database design imposing, 360
 - database performance, 347–356
 - creating indexes, 352–356
 - monitoring behavior, 347–348
 - using VACUUM command, 348–352
 - locking tables, 266
 - PEAR database abstraction interface, 460
 - PHP building SQL queries, 451
 - PostgreSQL, 11
 - transaction isolation, 255
 - deadlocks, 262
 - using DISTINCT keyword, 85
 - using subqueries, 188
- Perl
 - accessing PostgreSQL from, 465–489
 - code beginning with symbols (\$, @, %), 473
 - installing Perl DBI, 468–472
 - installing Perl modules, 466–468
 - resources for learning, 466
 - ways to access PostgreSQL from, 465
- perl (with-perl) option, 51
- Perl DBI
 - architecture, 468
 - available_drivers function, 483
 - binding parameters, 481–483
 - creating XML from DBI queries, 485–489
 - creating SQL from XML, 488–489
 - creating XML from SQL, 487–488
 - data access with PostgreSQL, 15
 - data_sources function, 483

- database connections
 - connecting to PostgreSQL, 476
 - connection attributes, 475
 - DBI environment variables, 475
 - specifying Data Source Names (DSNs), 474
- database connections, 473
- DBD (Database Driver) modules, 468, 469
- DBD::CSV driver, 469
- DBI environment variables, 475
- DBI features, 483
- DBI module, 469
- DBIx modules, 484
- DBIx::Easy, 484–485
- DBIx::XML_RDB module, 485
- executing SQL, 477–478
- installing DBI, 468–472
 - and PostgreSQL DBD from source, 471
 - and PostgreSQL DBD on Windows, 469
 - installing DBD::ODBC, 470
 - installing DBD::PgPP, 470
 - installing DBI bundle, 470
- result sets, 478–481
- using, 472–483
 - database connections, 477
- permissions
 - Data Control Language (DCL), 9
 - granting PostgreSQL connection permissions, 54
 - listing, 119
 - listing access permissions, 121
 - psql complaining about `pg_shadow`, 75
- persistent database connections
 - overuse of, 449
 - PHP making, 448
- Pg module
 - see* DBD::Pg database driver
- `pg_affected_rows` function, 453
- `pg_client_encoding` function, 459
- `pg_close` function, 449
- `pg_config` command, 52
- `pg_config` utility, 311
- `pg_connect` function, 447
- `pg_ctl` file, 310
- `pg_ctl` utility
 - options, 319
 - PostgreSQL files, 48
 - PostgreSQL installation, 47
 - postmaster process control, 56
 - starting and stopping server on Linux and UNIX, 319
 - stopping PostgreSQL, 58
- `pg_dbname` function, 449
- `pg_dump` utility
 - bin directory, 311
 - creating backup, 339
 - options, 340
 - pgAdmin III tool interfacing, 128
- `pg_dumpall` utility, 311
- `pg_fetch_array` function, 455
- `pg_fetch_object` function, 456
- `pg_fetch_result` function, 453
- `pg_fetch_row` function, 455
- `pg_field_xyz` functions, 457
- `pg_free_result` function, 457
- `pg_hba.conf` file
 - authentication methods, 312
 - data directory, 311
 - default configuration line, 312
 - Linux/UNIX installation, 54
 - Windows installation, 64
- `pg_host` function, 449
- `pg_ident.conf` file, 311, 313
- `pg_last_error` function, 458, 452
- `pg_last_notice` function, 458
- `pg_locks` view, 348
- `pg_num_fields` function, 453
- `pg_num_rows` function, 449
- `pg_options` function, 449
- `pg_pconnect` function, 448
- `pg_port` function, 449
- `pg_proc` table, 273
- `pg_query` function, 452
- `pg_restore` utility
 - bin directory, 311
 - recovering database, 342
- `pg_stat_activity` function, 348
- `pg_tty` function, 449
- PG_VERSION file, 311
- `pg_xyz` tables, 77
- pgAdmin III tool, 125–129
 - adding server connection in, 127
 - Backup dialog box, 129
 - backups, 128
 - database backup and recovery, 343–346
 - examining table properties with, 128
 - features, 125
 - installing, 125

- installing PostgreSQL on Windows, 60
- listing schemas, 332
- managing users with, 324
- order information viewed in, 29
- PostgreSQL configuration methods, 321
- restores, 128
- semicolon, using, 26
- using, 126
- vacuuming from pgAdmin III, 352
- viewing customer data, 23
- PGconn
 - database connections using libpq, 387
 - libpq connection parameters, 391
- PGDATA environment variable, 319
- PGDATABASE environment variable, 114
- PGDATESTYLE environment variable, 95
- pgFoundry (pgfoundry.org), 146
- PGHOST environment variable, psql
 - specifying database server host, 118
 - starting psql, 114
- pgmonitor, 147
- pgport (with-pgport) option, 51
- PGPORT environment variable, psql
 - specifying database server port, 118
 - starting psql, 114
- PgPP DBD database driver
 - see* DBD::Pg database driver
- PGRES_BAD_RESPONSE status, 393
- PGRES_COMMAND_OK status, 393
- PGRES_EMPTY_QUERY status, 393
- PGRES_FATAL_ERROR status, 393
- PGRES_NONFATAL_ERROR status, 393
- PGRES_POLLING_FAILED status, 415
- PGRES_POLLING_OK status, 415
- PGRES_TUPLES_OK status, 393
- PGSQL_XYZ argument values, 455
- PGUSER environment variable, psql
 - specifying database username, 119
 - starting psql, 114
- phantom reads
 - transaction isolation, 258
- phone numbers
 - choosing data type, 376
- PHP
 - accessing PostgreSQL from, 445–463
 - adding PostgreSQL support, 445–446
 - building queries, 450–452
 - creating complex queries, 451
 - executing queries, 452
 - checking for PostgreSQL support in, 445, 446
 - described, 445
 - error handling, 458–459
 - functions and result sets, 452–458
 - extracting values from result sets, 453
 - freeing result sets, 457
 - getting field information, 456
 - type conversion of result values, 458
 - making database connections, 447–450
 - closing connection, 449
 - connection information functions, 449
 - creating new connection, 447–448
 - creating persistent connection, 448–449
 - example script, 447
 - PostgreSQL connection parameters, 448
 - quotation mark usage, 447
 - retrieving connection handle
 - information, 449
 - using PHP variables, 447
 - object-orientation extensions for, 459
 - resources for information, 446
 - resources for learning, 445
 - support for character encoding, 459
 - using PEAR, 459–463
 - database abstraction interface, 460
 - using PHP API for PostgreSQL, 446–447
- php.ini file
 - levels of error reporting, 458
- phpPgAdmin tool, 129–133
 - browsing table data, 132
 - data import functionality, 132, 133
 - functionality, 129
 - installing, 130
 - phpPgAdmin login, 131
 - phpPgAdmin main page, 131
 - using, 130
- pipes
 - sending query output to filename, 118
- pl (PostgreSQL-pl) binary package, 45
- PL/pgSQL, 277–298
 - case sensitivity, 282
 - dynamic queries, 297
 - execution control structures, 289
 - conditional statements, 291
 - loops, 292
 - function arguments, 283
 - function declarations, 284
 - keywords, 285
 - requirements to use, 277
 - stored procedures, 282–298

- placeholders
 - binding parameters, Perl DBI, 481
- platforms, 11
- point data type, 209, 548
- point-in-time recovery, 346
- polygon data type, 209, 548
- Pooling option
 - Npgsql in Mono, 521
- pop_tablename.sql file, 69
- port connection parameter/option
 - pg_connect function, 448
 - PgPP driver, 474
 - PQconnectdb function, 388
- Port/port options
 - Npgsql in Mono, 521
 - PostgreSQL.conf file, 314
- portability
 - JDBC drivers, 492
- ports
 - setting TCP port number, 316
 - specifying database server port, 118, 330
 - specifying port, 322
- position function, 275
- postgres file, 310
- postgres user
 - creating, 53
 - creating first database, 114
 - data access, 53
 - default database installation, 317
 - initializing database, 54
 - security caution, 54
 - limiting use of user id, 74
 - running initdb utility, 54
 - secure PostgreSQL installations, 317
- PostgreSQL
 - accessing from C using embedded SQL, 419–444
 - accessing from C using libpq, 385–417
 - accessing from C#, 517–541
 - Npgsql in Mono, 520–539
 - Npgsql in Visual Studio, 539
 - ODBC .NET data provider on Windows, 517–520
 - accessing from Java, 491–516
 - accessing from Perl, 465–489
 - accessing from PHP, 445–463
 - building queries, 450–452
 - error handling, 458–459
 - functions and result sets, 452–458
 - making database connections, 447–450
 - support for character encoding, 459
 - using PEAR, 459–463
 - using PHP API for PostgreSQL, 446–447
 - adding resources to execution path, 58
 - architecture, 13, 14
 - checking if PostgreSQL running, 64
 - client programs connecting, 14
 - commands, 551–572
 - commercial support, 13
 - copyright, 15
 - described, 11
 - exiting back to shell, 66
 - features, 11
 - history of, 12
 - internal configuration, 320–338
 - configuration methods, 320–321
 - database management, 328–331
 - group configuration, 325–326
 - privilege management, 337–338
 - schema management, 331–337
 - tablespace management, 326–328
 - user configuration, 321–324
 - official PostgreSQL site, 16
 - official releases, 13
 - platforms, 11
 - programming language, 1
 - releases, 13
 - reliability, 12
 - resources, 16
 - for PostgreSQL tools, 146
 - scalability, 13
 - showing usage and distribution terms, 119
 - source code for, 49
 - upgrading, 46
 - Windows versions supported, 59
- PostgreSQL database server
 - see* databases servers
 - see also* servers
- PostgreSQL installation on Linux/UNIX, 43–58
 - building from source code, 43
 - data directory, 47
 - data storage for different file categories, 48
 - Debian Linux users installing, 44
 - drawback of single directory, 47, 48
 - from Linux binaries
 - binary packages, 44
 - downloading binaries, 44

- from Linux binaries, 44
- from SuSE packages with YaST2, 46
- installing RPM Package Manager, 45
- installing from source code, 49–52
 - compiling from source code, 50
 - configure script, 50
 - GNU tools, 50
 - make command, GNU, 52
 - querying configuration, 52
- setting up PostgreSQL, 53–58
 - configuring automatic postmaster startup, 57
 - connecting to database, 56
 - creating database directory, 53
 - creating postgres user, 53
 - granting connection permissions, 54
 - initializing database, 53
 - starting postmaster process, 55
 - stopping PostgreSQL, 58
- using KPackage, 48, 49
- PostgreSQL installation on Windows, 59–64
 - configuring client access, 64
 - using Windows installer, 59
 - database initialization, 62
 - installation options, 60
 - procedural languages, 63
 - processes, 63
 - program menu, 64
 - service configuration, 61
 - setup recommendations, 61
 - Windows versions supported, 59
- PostgreSQL.conf file, 311, 313, 314
- PostgreSQL/PostgreSQL-xyz binary packages, 44
 - YaST2 installation tool, 45
- PostgreSQLMetaData.java class, 500
- postmaster application, 53
 - checking if PostgreSQL running, 64
 - checking if running, 56
 - ensuring automatically run at startup, 57
 - location of database files, 55
 - pg_ctl utility controlling, 56
 - PostgreSQL files, 48
 - PostgreSQL installation, 47
 - process ID, 311
 - running processes on Linux and UNIX, 318
 - starting, 55, 56, 318
 - starting and stopping server on Linux and UNIX, 319
 - stopping, and restarting, 56
 - postmaster file, 310
 - postmaster.opts file, 311, 315, 316
 - postmaster.pid file, 311
 - pow function, 274
 - PPM (Perl Package Manager) utility
 - installing Perl modules, 467–468
 - PQbinaryTuples function, libpq, 411
 - PQcancelRequest function, libpq, 415
 - PQclear function, libpq, 392
 - PQcmdTuples function, libpq, 397, 399
 - PQconnectdb function, libpq, 387, 388
 - PQconnectPoll function, libpq, 415
 - PQconnectStart function, libpq, 415
 - PQconsumeInput function, libpq, 414
 - PQerrorMessage function, libpq
 - asynchronous working, 412
 - connection errors, 391
 - executing SQL with libpq, 394
 - PQexec function, libpq
 - executing SQL queries with, 394
 - executing SQL with libpq, 392, 397
 - fetching results using cursors, 405, 408
 - including user-specified data in, 396
 - managing transactions, 404
 - PQfinish function, libpq
 - closing connection using, 389
 - NULL pointers, 391
 - PQflush function, libpq, 414
 - PQfname function, libpq, 397
 - PQfnnumber function, libpq, 398
 - PQfsize function, libpq, 398
 - PQgetisnull function, libpq, 401, 438
 - PQgetlength function, libpq, 399
 - PQgetResult function, libpq, 412
 - PQgetvalue function, libpq, 399, 400
 - PQisBusy function, libpq, 414
 - PQisnonblocking function, libpq, 413
 - PQnfields function, libpq, 397
 - PQntuples function, libpq
 - executing SQL with libpq, 397
 - fetching results using cursors, 408
 - PQprint function, libpq, 401
 - PQprintOpt structure, libpq, 401
 - PQreset function, libpq, 389
 - PQresStatus function, libpq, 394
 - PQresultErrorMessage function, libpq, 394, 399
 - PQresultStatus function, libpq, 392, 393
 - PQsendQuery function, libpq, 412
 - PQsetnonblocking function, libpq, 412

- PQsocket function, libpq, 414
- PQstatus function, libpq
 - checking state of connection, 389
 - NULL pointers, 391
- precedence
 - operator precedence, 269, 270
- Precision property
 - NpgsqlParameter class, 533
- predicate calculus, 8
- prefix option
 - configure script, 51
- PREPARE command, 568
- Prepare method
 - NpgsqlCommand class, 526
- prepare method, PEAR, 462
- prepareCall method
 - Connection interface, java.sql, 499
- prepared statements
 - JDBC statements, 507, 512
 - using prepared statements, 516
 - using with parameters, Npgsql in Mono, 535–536
- PreparedStatement interface, java.sql, 507
 - clearParameters method, 514
 - execute method, 513
 - executeQuery method, 513
 - executeUpdate method, 513
 - executing prepared SQL statements, 513
 - setBoolean method, 514
 - setInt method, 514
 - setString method, 514
 - updating data, 513
 - using prepared statements, 512
 - writing JDBC client using, 514,–516
- PreparedStatementClient class, 514
- prepareStatement method
 - Connection interface, java.sql, 499
- preprocessors
 - ecpg preprocessor, 419
 - Oracle and Informix, 420
- previous method
 - ResultSet interface, java.sql, 503
- primary keys
 - establishing in design, 372–373
 - identifying rows uniquely, 22
 - PRIMARY KEY constraint, 219
 - column constraints, 218
 - table constraints, 222
- RDBMS, 7
- PrintError attribute, Perl DBI, 475
- printing
 - printing rows only, 119
 - setting printing option, 118
- println method
 - DriverManager class, java.sql, 496
- PrintWriter method
 - DriverManager class, java.sql, 496
- privileges
 - GRANT command, 566
 - privilege management, 337–338
 - grantable privileges, 337
 - granting privileges, 337
 - revoking privileges, 338
 - REVOKE command, 568
- procedural languages, 276–282
 - ALTER LANGUAGE, 554
 - CREATE LANGUAGE, 560
 - DROP LANGUAGE, 564
 - installing handler, 277
 - installing PostgreSQL on Windows, 63
 - LANGUAGE clause, 276
 - PL/pgSQL, 277–282
 - programming languages, 28
 - requirements for handling, 277
- procedures
 - defining trigger procedure, 300
 - trigger procedure variables, 303
- process function
 - DBI::Easy module, 485
- process ID (PID)
 - postmaster application, 311
- processes
 - installing PostgreSQL on Windows, 63
 - running processes on Linux and UNIX, 318
- program menu
 - installing PostgreSQL on Windows, 64
- programming languages, 1
 - declarative languages, 28
 - procedural languages, 28
- projection
 - accessing data, 27
- prompt changes
 - continuation lines, 193
 - starting up psql on Windows, 75
- prompts, createuser utility
 - prompting for password, 323

- prompts, psql
 - continuation prompt, 108
 - indicating input expected, 115
 - indicating permissions, 114
 - prompting for password, 119
 - single-line mode, 115
 - Protocol option
 - Npgsql in Mono, 521
 - pset command (\pset), psql, 120, 575
 - setting printing option, 118
 - pseudo users
 - data access, 53
 - psql command-line tool, 113–121
 - basic commands, 78
 - bin directory, 310
 - command history, 115
 - command line options, 118–119, 573–574
 - command syntax, 118
 - command types, 114
 - commands
 - describing table, 150
 - executing commands from a file, 78
 - getting a help message, 78
 - getting help on specified command, 78
 - listing data types, 78
 - listing operators, 78
 - listing tables, 78
 - never splitting over lines, 115
 - quitting psql, 78
 - resetting buffer, 78
 - continuation prompt, 108
 - copy command, 159–162
 - CREATEDB option, 75
 - creating database, 75
 - creating user, 75
 - currval function, 156
 - deleting database, 75
 - deleting user, 75
 - examining database structure, 117
 - exiting psql, 75
 - internal commands, 114, 119–121, 574–576
 - issuing commands in psql, 114–115
 - max function, 156
 - nextval function, 155
 - prompts, 114, 115
 - psql complaining about pg_shadow, 75
 - resolving startup problems, 75–77
 - scripting psql, 115–117
 - creating and populating tables, 115
 - executing file, 116
 - reading psql commands from file, 115
 - redirecting query output, 117
 - simple reports, 116
 - semicolon, 26, 79, 161
 - sequence numbers, 161
 - setting psql variable, 119
 - setval function, 156
 - SHOW command, 97
 - simple administrative tasks, 56
 - SQL commands, 114
 - sql extension to files, 116
 - starting psql, 114
 - starting up on Linux, 74
 - starting up on Windows, 74–75
 - choosing database for connection, 75
 - connecting to remote server, 75
 - identifying user, 75
 - switching user, 74
 - terminating SQL statements in, 79
 - using, 74–78
 - viewing customer data, 23
 - psql option
 - installing PostgreSQL on Windows, 60
 - psqlodbc
 - ODBC drivers, 121
 - PUBLIC keyword
 - GRANT command, 337
 - public schema, 331
 - pwd option (-pwd)
 - sql2xml.pl script, 487
 - xml2sql.pl script, 488
 - python (PostgreSQL-python) binary
 - package, 45
 - python (with-python) option, 51
- ## Q
- q command (\q), psql, 78, 120, 575
 - internal commands, psql
 - q option (-q)
 - createdb/dropdb utilities, 330
 - createuser utility, 322
 - psql, 574
 - rpm query, 48
 - qecho command (\qecho), psql, 120, 575

queries

- see also* SQL queries
 - clearing query buffer, 115
 - dynamic queries, 297
 - editing query buffer, 115
 - examining query buffer, 115
 - executing SQL with libpq, 397, 392–401
 - asynchronous working, 412, 415
 - PHP building, 450–452
 - preparing and executing with PEAR, 462
 - query language, 8
 - RDBMS, 8
 - reading and executing from file, 120
 - redirecting query output, psql, 117
 - Recall building queries, 136
 - repeating a query, 115
 - responsibilities of DBMS, 10
 - running single query, psql, 118
 - sending query output to filename, 118
 - sending to back-end, 120
 - subqueries
 - see* subqueries
 - viewing query history, 115
 - writing query buffer to file, 121
- query buffer
- editing, 120
 - examining, 115
 - resetting, 120
 - showing content of, 120
- query optimizer, 349
- query output
- printing with libpq, 401–403
 - running quietly, 118
 - sending to file or pipe, 120
 - writing text to query output stream, 120
- query plan, 349
- quiet option (-quiet)
- createuser utility, 322
- quitting psql, 78, 120
- quotes
- dollar quoting, 282
 - using in stored procedure creation, 281
- R**
- r command (\r), psql, 78, 120, 576
- query history, 115
- R option (-R), psql, 118, 574
- radians function, 274

RAISE statement

- execution control structures, 290
- RaiseError attribute, Perl DBI, 475
- random function, 274
- RDBMS (relational database management system)
- atomicity, 7
 - Codd, E.F., 6
 - data integrity, 6
 - database models, 6
 - PostgreSQL, 11
 - predicate calculus, 8
 - primary keys, 7
 - queries, 8
 - records, 6
 - responsibilities of DBMS, 10
 - SQL, 8
 - tuples, 6
- Read Committed isolation level, 260, 261
- Read method
- NpgsqlDataReader class, 528
- Read Uncommitted isolation level, 260, 261
- readability of SQL
- using subqueries, 188
- reading data
- dirty reads, 256
 - phantom reads, 258
 - unrepeatable reads, 257
- readline
- preventing line editing, 118
- real data type, 546
- records
- flat files for data storage, 2
 - setting record separator, 118
 - tuples, 6
- RecordsAffected property
- NpgsqlDataReader class, 527
- recovery
- database backup and recovery, 338–346
 - point-in-time recovery, 346
- recursive relationship pattern
- database design, 382–383
- Red Hat
- packages for download available at, 44
- Red Hat Database, 16
- Red Hat Package Manager
- see* RPM

- REFERENCES keyword
 - column constraints, 218
 - foreign key constraint, 233, 234
 - table constraints, 222
- REFERENCES privilege
 - grantable privileges, 337
- referential integrity
 - responsibilities of DBMS, 11
- refreshRow method
 - ResultSet interface, java.sql, 506
- registerDriver method
 - DriverManager class, java.sql, 494
- regular expression matches
 - operators performing, 268
 - string operators, 272
- REINDEX command, 568
- reject authentication methods
 - pg_hba.conf file, 312
- Recall, 133–137
 - browsing a table with, 135
 - building queries, 136
 - connecting to database, 134
 - creating forms, 135
- relating data entities
 - cardinality symbols, 367
 - database design, 359, 366
- relating tables
 - additional WHERE clause conditions, 102
 - SQL92 SELECT syntax, 110
 - creating simple database design, 36, 37
 - identifying foreign keys, 373–375
 - linking tables, 101
 - relating three or more tables, 106–110
 - database schema, 106
 - joining multiple tables, 108
 - using DISTINCT keyword, 110
 - relating two tables, 100–105
 - specifying column and table, 102
 - using joins, 29
- relational database management system
 - see RDBMS
- relative method
 - ResultSet interface, java.sql, 503
- RELEASE SAVEPOINT command, 568
- releases, PostgreSQL, 13
- reload option, pg_ctl utility, 319
- remote access
 - granting connection permissions, 54
- RENAME declaration, 286
- RENAME TO option
 - ALTER USER command, 323
- reorders function
 - creating stored procedures, 295
- Repeatable Read isolation level, 260, 261
- repeating groups
 - database design, 34
 - flat files problem, 3
- reports
 - Microsoft Access creating, 141
 - scripting psql, 116
- requirements
 - database design supporting, 359
- reserved keywords
 - name, 362
- RESET command, 568
- resolving problems, 75–77
- resources, 16
 - for PostgreSQL tools, 146
- responses
 - quiet option (-quiet), 322
- restart option, pg_ctl utility, 319, 320
- restores
 - pgAdmin III tool, 128
 - utility to restore database, 311
- result sets
 - JDBC result sets, 502–507
 - accessing result set data, 504
 - concurrency types, 502
 - methods for traversing, 503
 - result set types, 502
 - Statement interface querying, 509
 - updateable result sets, 505
 - Perl DBI and, 478–481
 - fetching results, 479
 - statement handle attributes, 479
 - PHP functions and, 452–458
 - extracting values from, 453
 - freeing result sets, 457
 - getting field information, 456
 - type conversion of result values, 458
- ResultSet interface, java.sql
 - absolute method, 503
 - accessing result set data, 504
 - afterLast method, 504
 - beforeFirst method, 504
 - cancelRowUpdates method, 506
 - close method, 507
 - deleteRow method, 505

- first method, 503
 - getBoolean method, 504
 - getConcurrency method, 502
 - getFetchDirection method, 504
 - getFetchSize method, 504
 - getInt method, 504
 - getMetaData method, 507
 - getString method, 504
 - getting concurrency type, 502
 - getting result set type, 502
 - getType method, 502
 - insertRow method, 507
 - isAfterLast method, 503
 - isBeforeFirst method, 503
 - isFirst method, 503
 - isLast method, 503
 - last method, 503
 - moveToCurrentRow method, 506
 - moveToInsertRow method, 506
 - next method, 503
 - previous method, 503
 - refreshRow method, 506
 - relative method, 503
 - rowDeleted method, 505
 - rowUpdated method, 506
 - setFetchDirection method, 504
 - traversing result sets
 - manipulating fetch direction and size, 504
 - querying cursor position, 503
 - scrolling result sets, 503
 - updateable result sets
 - deleting data, 505
 - inserting data, 506
 - updating data, 506
 - updateBoolean method, 506
 - updateInt method, 506
 - updateRow method, 506
 - updateString method, 506
 - retrieving data
 - see* data access
 - RETURN statement
 - execution control structures, 290
 - REVERSE option
 - FOR loops, 294
 - REVOKE command, 568
 - privilege management, 338
 - RIGHT OUTER JOIN keywords, 199
 - ROLLBACK command, 569
 - single user transactions, 248
 - transaction examples, 248, 249, 251
 - transactions, 244
 - rollback method
 - Connection interface, java.sql, 499
 - ROLLBACK TO SAVEPOINT command, 251, 569
 - round function, 214, 274
 - rowDeleted method
 - ResultSet interface, java.sql, 505
 - rows
 - see* database rows
 - rowtype declaration syntax, 287
 - SELECT INTO statement, 289
 - rowUpdated method
 - ResultSet interface, java.sql, 506
 - RPM (RPM Package Manager)
 - installing on Linux, 45
 - list option (-l), 48
 - listing installed files, 48
 - packages for download available at, 44
 - query option (-q), 48
 - upgrading PostgreSQL, 46
 - RULE privilege
 - grantable privileges, 337
 - rules
 - CREATE RULE, 560
 - DROP RULE, 564
 - runtime parameters
 - SET command, 570
- S**
- s command (\s), psql, 120, 576
 - query history, 115
 - S option (-S)
 - pg_dump utility, 341
 - s option (-s)
 - pg_ctl utility, 319
 - pg_dump utility, 341
 - pg_restore utility, 342
 - psql, 118, 574
 - S option (-S), psql, 119, 574
 - splitting commands over lines, 115
 - sample database, creating
 - see under* database creation
 - SAVEPOINT command, 569
 - RELEASE SAVEPOINT, 568
 - ROLLBACK TO, 251, 569
 - transactions, 244, 252

- scalability, PostgreSQL, 13
- Scale property
 - NpgsqlParameter class, 533
- schema
 - see* database schema
- scripts, psql
 - i command (\i) running, 67
 - scripting database schema, 67
 - scripting psql, 115–117
- scrollable result sets, 502
 - ResultSet interface, java.sql, 503
- search_path option
 - PostgreSQL.conf file, 315
- second normal form
 - database design, 379
- security
 - installing PostgreSQL on Windows, 62
 - limiting use of postgres user, 74
 - Npgsql in Mono, 521
 - PostgreSQL installations, 317
 - reasons for using stored
 - procedures/triggers, 306
 - responsibilities of DBMS, 11
- SELECT privilege
 - grantable privileges, 337
- SELECT statement, 570
 - accessing data, 73–112
 - advanced features, 173–200
 - aggregate functions
 - see* aggregate functions, SELECT
 - AND conditional operator, 88
 - BETWEEN keyword, 89, 90
 - cast function, 86
 - column aliases, 81
 - counting number of rows selected, 31
 - dates and times, 94–100
 - DISTINCT keyword, 84, 85
 - duplicated rows returned, 83
 - suppressing duplicates, 84, 85
 - executing SQL with libpq, 397
 - FROM clause, 78
 - functions useable in, 273
 - IN keyword, 89
 - joins, 192–200
 - LIKE keyword, 91
 - LIMIT keyword, 92
 - listing tables in schema, 336
 - multiple tables, selecting from, 100–110
 - multiplying results, 86
 - no data returned, error handling, 428
 - NULLs, checking for, 93–94
 - OFFSET clause, 92
 - operators outside WHERE clause, 269
 - OR conditional operator, 89
 - ORDER BY clause, 81, 82
 - default sort order, 82
 - using ASC and DESC, 82, 83
 - pattern matching, 91
 - performing calculations in, 86
 - PostgreSQL group configuration, 326
 - PostgreSQL user configuration, 323
 - relating tables
 - JOIN ... ON clause, 111
 - joining multiple tables, 108
 - relating three or more tables, 106–110
 - relating two tables, 100–105
 - using DISTINCT keyword, 110
 - retrieving data with ecpg, 436
 - SELECT INTO, 288, 570
 - selecting all columns, 79
 - selecting named columns, 80, 81
 - simple SELECT, 78–85
 - SQL92 SELECT syntax, 110–112
 - subqueries
 - see* subqueries
 - table name aliases, 105
 - using asterisk (*), 79
 - WHERE clause, 87–93
- selectall_arrayref function, Perl DBI, 479
- selection
 - accessing data, 26
- selectrow_array function, Perl DBI, 479
- self joins, 194–196
 - database design, 383
- semicolon
 - pgAdmin III tool using, 26
 - psql command-line tool, 26, 79, 115, 161
- sentinel value, 41
- separators
 - setting record separator, 118
- sequence diagrams
 - JDBC client connecting, 496
- sequence numbers
 - see also* serial data type
 - accessing, 155–157
 - currval function, 156
 - nextval function, 155
 - setval function, 156

- copy command, psql, 161
- currval function, psql, 161
- naming conventions, 155
- out-of-step sequence problem, 155
- sequences
 - ALTER SEQUENCE, 554
 - CREATE SEQUENCE, 561
 - DROP SEQUENCE, 565
 - dropping sample database tables, 69
 - listing, 119
- serial data type, 207, 548, 549
 - see also* sequence numbers
 - currval function, psql, 162
 - data types, 40
 - identifying rows uniquely, 23
 - incrementing, 40
 - inserting data into serial columns, 154–157
 - providing values for serial data, 155
 - loading data from another application, 163
 - psql table description, 150
- Serializable isolation level, 260, 261
- server (PostgreSQL-server) binary package, 44
- server control, 318–320
- Server option
 - Npgsql in Mono, 521
- servers
 - see also* database servers
 - adding server connection in pgAdmin, 127
 - specifying database server host/port, 118
 - utility to start, stop, and restart, 310
- ServerVersion property
 - NpgsqlConnection class, 522
- service configuration
 - installing PostgreSQL on Windows, 61
- sessions
 - SET SESSION AUTHORIZATION, 571
- SET clause/command
 - datestyle, 95, 97
 - syntax, 570
 - UPDATE statement, 168
- set command (\set), psql, 121, 576
- SET CONSTRAINTS command, 571
- SET SESSION AUTHORIZATION
 - command, 571
- SET statement
 - EXEC SQL syntax, 427
- SET TRANSACTION command, 571
- setAutoCommit method
 - Connection interface, java.sql, 499
- setBoolean method
 - PreparedStatement interface, java.sql, 514
- setErrorHandler() method
 - PEAR_Error object, 462
- setFetchDirection method
 - ResultSet interface, java.sql, 504
- setInt method
 - PreparedStatement interface, java.sql, 514
- setLoginTimeout method
 - DriverManager class, java.sql, 496
- setLogWriter method
 - DriverManager class, java.sql, 496
- setString method
 - PreparedStatement interface, java.sql, 514
- setTransactionIsolation method
 - Connection interface, java.sql, 500
- setval function, psql
 - sequence numbers, 156, 157, 161
- share directory
 - PostgreSQL installation, 47
 - system configuration, 316
- shared locks, 262
- shared memory segments
 - utility to delete, 311
- shared_buffers option
 - PostgreSQL.conf file, 314
- shell
 - escape to or execute command, 121
- shift left operator, 271
- shift right operator, 271
- SHOW command, psql, 571
 - datestyle variable, 97
- shutdown mode, 319
- silent mode, setting, 319
- sin function, 275
- single user transactions
 - see under* transactions
- single-line comments, 284
- single-line mode, setting, 119
- single-step mode, setting, 118
- Size property
 - NpgsqlParameter class, 533
- smallint data type, 207, 546
- sn option (-sn)
 - sql2xml.pl script, 487
 - xml2sql.pl script, 488
- source code
 - installing PostgreSQL on Linux/UNIX
 - from, 49–52

- spreadsheets
 - data storage limitations, 17–20
- sprintf function
 - PHP building SQL queries, 450
- SQL
 - see also* embedded SQL
 - client processing large amounts of data, 404
 - command keywords
 - case sensitivity, 10
 - command types, 9
 - creating SQL from XML, 488–489
 - creating XML from SQL, 487–488
 - definition, 8
 - executing SQL
 - using Perl DBI, 477–478
 - executing SQL with libpq, 392–401
 - binary values, 411
 - determining query status, 392–394
 - executing queries with PQexec, 394–396
 - extracting data from query results, 397–400
 - handling NULL results, 400
 - managing transactions, 404
 - updating or deleting rows, 396
 - user specified data in query, 396
 - using cursors, 404–411
 - introduction, 9–10
 - JDBC API SQL exceptions and warnings, 494
 - levels of conformance, 8
 - PostgreSQL configuration methods, 320
 - variations, 8
- SQL batches
 - java.sql.Statement interface, 509
- SQL commands
 - PostgreSQL commands, 551
 - PostgreSQL syntax for, 552–572
 - psql command types, 114
- SQL control area
 - see* sqlca
- sql extension to files (.sql)
 - executing files, 116
- SQL functions, 298–299
 - CREATE FUNCTION, 298
 - parameters, 298
 - returning multiple rows, 298
- SQL queries
 - PHP building queries, 450
 - building complex queries, 451
 - executing queries, 452
- SQL transactions
 - see* transactions
- sql2xml.pl script, 487
- SQL92 SELECT syntax, 110–112
 - JOIN ... ON clause, 111
 - outer joins, 198
- sqlca (SQL control area)
 - error codes, 429
 - include directive, 430
 - reporting embedded SQL errors, 428–430
 - reporting rows affected, 429
 - resetting sqlca structure, 428
 - trapping embedded SQL errors, 431
- sqlcode field, sqlca, 428
- sqlerrd field, sqlca
 - fetching rows into cursor, 441
 - handling empty results, 440
 - reporting SQL errors, 429
- sqlerrm field, sqlca, 428
- sqlerror condition
 - whenever statement, EXEC SQL, 431
- SQLException class
 - JDBC API, 494
- sqlprint action
 - whenever statement, EXEC SQL, 431
- sqlwarn array, sqlca, 429
- SQLWarning class
 - JDBC API, 494
- sqlwarning condition
 - whenever statement, EXEC SQL, 431
- sqrt function, 274
 - function arguments, 283
- square root operator, 271
- SSL option
 - Npgsql in Mono, 521
- standard deviation functions, 174
- start option, pg_ctl utility, 319
- START TRANSACTION command, 244, 571
- startup
 - configuring automatic startup, 57
 - not reading startup file, 119
 - reading startup file, psql, 114
- state
 - libpq checking state of connection, 389
- State property
 - NpgsqlConnection class, 522
- statement handle attributes
 - Perl DBI and result sets, 479

- Statement interface, java.sql, 507
 - addBatch method, 510
 - clearBatch method, 510
 - execute method, 509
 - executeBatch method, 510
 - executeQuery method, 508
 - executeUpdate method, 509
 - executing statements, 508
 - getMoreResults method, 509
 - getResultSet method, 509
 - getUpdateCount method, 509
 - handling SQL batches, 509
 - querying result sets, 509
 - using statements, 508
 - writing JDBC client using, 510
- statement_timeout option
 - PostgreSQL.conf file, 315
- StatementClient class
 - writing JDBC client using, 510
- statements
 - DEALLOCATE command, 563
 - EXECUTE command, 566
 - EXPLAIN command, 566
 - JDBC clients using, 498
 - JDBC statements, 507–516
 - PREPARE command, 568
 - using with parameters, Npgsql in Mono, 533–534
- statistics
 - ANALYZE command, 556
 - commands being executed, 315
 - setting to collect, 315
 - updating optimizer statistics, 349
 - viewing PostgreSQL statistics, 348
- stats_command_string option
 - PostgreSQL.conf file, 315
- stats_start_collector option
 - PostgreSQL.conf file, 315
- status option, pg_ctl utility, 319, 320
- stock table
 - creating table, 68, 579
 - populating sample database tables, 71
 - schema for item and stock tables, 196
- stop option, pg_ctl utility, 319, 320
- stopping PostgreSQL, 58
- stored procedures, 282–298
 - see also* functions
 - ALIAS declaration, 285
 - assignments, 288
 - automated re-ordering example, 295
 - comments, 284
 - creating, 281
 - declarations, 284
 - dynamic queries, 297
 - execution control structures, 289
 - function arguments, 283
 - grantable privileges, 337
 - PL/pgSQL, 282–298
 - PostgreSQL programming language support, 62
 - reasons for using, 306
 - RENAME declaration, 286
 - SQL functions, 298–299
 - triggers, 299–306
 - variable declarations, 286, 287
- storing data
 - see* data storage
- string concatenation, 450
- string functions, 275
- string operators, 272
- strings
 - null-terminated strings
 - retrieving data with ecpg, 437
 - padding with spaces problem, 400
- strpos function, 214
- strtolower function, 450
- structure
 - database structure, 117
- structured text files
 - flat files compared, 3
- subdirectories
 - base directory, 310
 - data directory, 311
- subqueries, 185–192
 - correlated subqueries, 188–191
 - existence subqueries, 191–192
 - returning multiple rows, 187–188
- subsets of data
 - data columns, 27
 - data rows, 26, 27
- subsidiary table, 374
- substr function, 214
- substrng function, 275
- subtraction operator, 271
- sum function, 184
 - description, 174
 - DISTINCT keyword, 184
 - NULL values, 184

- superuser_reserved_connections option
 - PostgreSQL.conf file, 314
 - superusers
 - prompting for superuser password, 317
 - support, PostgreSQL, 13
 - surrogate keys
 - identifying rows uniquely, 23
 - SuSE Linux
 - data storage for different file categories, 48
 - packages for download available at, 44
 - system configuration, 309–316
 - base directory, 309
 - bin directory, 310–311
 - data directory, 311–316
 - doc directory, 316
 - include directory, 316
 - lib directory, 316
 - man directory, 316
 - share directory, 316
 - system tables, listing, 119
- T**
- T command (\T), psql, 121, 576
 - t command (\t), psql, 121, 576
 - T option (-T)
 - createdb/dropdb utilities, 330
 - psql, 119, 574
 - t option (-t)
 - ecpg arguments, 424
 - not specifying -t option, 424
 - pg_dump utility, 341
 - pg_restore utility, 342
 - psql, 119, 574
 - vacuumdb utility, 351
 - table option (-table)
 - sql2xml.pl script, 487
 - xml2sql.pl script, 488
 - table tag options
 - setting HTML table tag options, 119
 - tables
 - see* database tables
 - TABLESPACE option
 - CREATE DATABASE command, 329
 - tablespaces
 - ALTER TABLESPACE, 555
 - altering, 327
 - CREATE TABLESPACE, 562
 - creating, 327
 - DROP TABLESPACE, 565
 - dropping, 328
 - listing, 120
 - setting default for new database, 330
 - tablespace management, 326–328
 - tan function, 275
 - tcl (PostgreSQL-tcl) binary package, 45
 - tcl (with-tcl) option, 51
 - techdocs.PostgreSQL.org
 - resources for PostgreSQL tools, 147
 - TEMPLATE option
 - CREATE DATABASE command, 329
 - templates
 - creating first database, 114
 - specifying template database to copy, 330
 - temporal data type, 209, 547
 - temporary tables, 227–228
 - CREATE TABLE command, 217
 - test (PostgreSQL-test) binary package, 45
 - text
 - choosing data type, 376
 - writing to standard output, 120
 - text data type, 204, 547
 - support for large objects, 582
 - TG_XYZ trigger procedure variables, 303
 - third normal form
 - database design, 379
 - threads
 - scrollable result sets, 502
 - time
 - see* dates and times
 - time data type, 209, 547
 - time zone information, 209
 - Timeout option
 - Npgsql in Mono, 521
 - timeouts
 - JDBC API managing login timeouts, 496
 - setting deadlock timeout, 315
 - setting statement timeout, 315
 - setting time to complete authentication, 314
 - timestamp data type, 209, 547
 - component parts, 99
 - data types, 94
 - timestampz data type, 209
 - timezone files, 316
 - timezone option
 - PostgreSQL.conf file, 315
 - timing command (\timing), 121, psql, 576
 - turning on timing option, 354
 - to_char function, 275

- tools
 - resources for PostgreSQL tools, 146
- total sum of values in column
 - see* sum function
- tracking changes, 306
- transaction isolation levels
 - setting default, 315
 - undesirable phenomena, 256
- Transaction property
 - NpgsqlCommand class, 526
- transactions, 243–261
 - ABORT command, 552
 - ACID rules, 246–247
 - BEGIN ... COMMIT blocks, 244
 - BEGIN command, 556
 - CHECKPOINT command, 556
 - COMMIT command, 557
 - concurrent multiuser access to data, 244–246
 - double booking error, 244
 - definition, 244
 - END command, 565
 - esqlc program, writing, 420
 - explicit and implicit transactions, 261
 - isolation, 255–261
 - ANSI isolation levels, 260–261
 - changing isolation level, 261
 - dirty reads, 256–257
 - lost updates, 258–259
 - phantom reads, 258
 - unrepeatable reads, 257
 - JDBC handling database transactions, 499
 - locking, 262–266
 - deadlocks, 262–264
 - exclusive locks, 262
 - explicit locking, 264–266
 - shared locks, 262
 - log files, 247
 - logical unit of work, 244
 - managing with libpq, 404
 - multiple users, 255–261
 - nesting transactions, 254
 - ROLLBACK command, 244, 569
 - ROLLBACK TO SAVEPOINT command, 569
 - SAVEPOINT command, 244, 569
 - SET CONSTRAINTS command, 571
 - SET TRANSACTION, 571
 - single users, 247–255
 - multiple tables example, 250–251
 - savepoint example, 251–254
 - simple transaction example, 248–250
 - START TRANSACTION, 571
 - time to complete, 255
 - transaction size, 255
- TRIGGER privilege
 - grantable privileges, 337
- trigger_error function
 - error handling with PEAR, 462
- triggers, 299–306
 - AFTER triggers, 300
 - ALTER TRIGGER, 555
 - BEFORE triggers, 300
 - CREATE CONSTRAINT TRIGGER, 558
 - CREATE TRIGGER, 562
 - creating triggers, 300
 - deletion example, 303
 - updating example, 301
 - defining trigger procedure, 300
 - DROP TRIGGER, 565
 - dropping triggers, 300
 - procedure variables, 303
 - reasons for using, 306
- trigonometric functions, 274
- trim function, 214, 275
- troubleshooting
 - problems starting up psql, 75–77
 - psql complaining about pg_shadow, 75
- trunc functions, 274
- TRUNCATE command, 170–171, 571
 - see also* data handling; DELETE statement
- truncate operator, 271
- truncation
 - DataTruncation class, 494
- trust authentication mechanism
 - configuring client access, 64
 - granting PostgreSQL connection
 - permissions, 55
- trust authentication methods
 - pg_hba.conf file, 312
- tuples, RDBMS, 6
- type 1/type 2/type 3/type 4 JDBC drivers, 492, 493
- TYPE attribute
 - statement handles, Perl DBI, 480

TYPE column
 pg_hba.conf file, 312
 type conversion
 PHP result values, 458
 TYPE_FORWARD_ONLY result set type, 502
 TYPE_SCROLL_INSENSITIVE result set
 type, 502
 TYPE_SCROLL_SENSITIVE result set type, 502
 Types class, java.sql, 505
 types of data
see data types

U

U option (-U)
 createdb/dropdb utilities, 330
 createlang utility, 278
 createuser utility, 322
 creating user records, 65
 pg_dump utility, 341
 pg_restore utility, 342
 psql, 119, 574
 changing user, 117
 starting psql, 114
 rpm upgrade, 46
 vacuumdb utility, 351
 uid option (-uid)
 sql2xml.pl script, 487
 xml2sql.pl script, 488
 unaligned table output mode, psql, 118
 toggling between aligned and, 119
 unary arithmetic operators, 271
 operator precedence, 270
 uncorrelated subqueries, 188
 UNION join, 192–194
 including duplicates, 194
 UNION ALL join, 194
 UNIQUE option
 column constraints, 218, 219, 220
 CREATE INDEX command, 353
 NULLs, 219
 table constraints, 222
 uniqueness
 database design, rows, 33
 UNIX
 flat files for data storage, 2
 installing PostgreSQL on, 43–58
 running processes on, 318
 starting and stopping server on, 319
 UNLISTEN command, 572
 unrepeatable reads
 transaction isolation, 257
 unset command (\unset), psql, 121, 576
 UPDATE command, 572
 update function
 DBI::Easy module, 485
 UPDATE privilege
 grantable privileges, 337
 UPDATE statement, 165–168
see also updating data in database
 executing SQL with libpq, 396
 FROM option, 168
 importance of WHERE clause, 166, 167
 lost updates, 259
 ON UPDATE keyword, 241–242
 reporting rows affected, 429
 SET clause, 168
 transaction example, 250, 251
 triggers, 300, 302
 updating from another table, 168
 update*Boolean/~Int/~Row/~String* methods
 ResultSet interface, java.sql, 506
 updating data in database
see also UPDATE statement
 BatchUpdateException class, 494
 JDBC updateable result sets, 506
 lost updates, transaction isolation, 258
 PreparedStatement interface, java.sql, 513
 updating from another table, 168
 using count(*) syntax, 167
 using Npgsql in Mono, 536
 upgrading PostgreSQL, 46
 upper function, 275
 USER column
 pg_hba.conf file, 312
 user configuration
 PostgreSQL internal configuration,
 321–324
 user connection parameter/option
 pg_connect function, 448
 PQconnectdb function, 388
 User Id option
 Npgsql in Mono, 521
 user records, creating, 65
 useradd command, 53

users

- adding, Linux, 53
- allowing user to create new users, 323
- ALTER USER, 555
- authentication methods, 312
- changing user, psql, 117
- CREATE USER, 562
- creating user, 322
 - psql, 75
- deleting user, psql, 75
- DROP USER, 565
- handling multiuser database access, 25
- limiting use of postgres user, 74
- listing, 120, 323
- managing with pgAdmin III, 324
- modifying, 323
- removing, 324
- specifying user, 322
- specifying user ID number, 323
- specifying username to connect, 330
- utility to create database user, 310
- USING DELIMITERS option
 - copy command, psql, 160

V

v option (-v)

- ecpg arguments, 424
- pg_dump utility, 340
- pg_restore utility, 342
- vacuumdb utility, 351
- psql, 119, 574

V option (-V), psql, 574

VACUUM command

- database performance, 348–352
 - reclaiming space, 348
 - updating optimizer statistics, 349
 - vacuuming from command line, 351
 - vacuuming from pgAdmin III, 352
- syntax, 572
- VACUUM ANALYZE, 351, 353

vacuumdb utility

- bin directory, 311
- options, 351
- vacuuming from command-line, 351

VALID UNTIL option

- CREATE USER command, 322

validation, 306

Value property

- NpgsqlParameter class, 533

values

- NULL, 41
 - sentinel value, 41
- varchar data type, 204, 547
 - data types, 40
 - using host variables, 434
- variable declarations, 285, 286
 - composite variables, 287
 - record type, 288
 - CONSTANT modifier, 286
- examples, 287
- NOT NULL clause, 286
- rowtype declaration syntax, 287

variable names

- naming conventions, 285
- setting psql variable, 119

variance functions, 174

VERBOSE option

- VACUUM command, 349

version information

- configuration, 311
- showing version information, 119

version option

- pg_config command, 52

version option (-version)

- createdb/dropdb utilities, 330
- psql, 119

views

- see* database views

Visual Studio

- using Npgsql in Visual Studio, 539

W

w command (\w), psql, 121, 576

W option (-W)

- createdb/dropdb, 330
- createlang utility, 278
- initdb utility, 317
- pg_ctl utility, 319
- psql, 119, 574

w option (-w)

- pg_ctl utility, 319

WARNING exception level, 290

warnings

- JDBC API SQL warnings, 494
- SQLWarning class, 494

- whenever statement
 - database specific, 431
 - overuse of, 432
 - trapping errors, 431
 - using host variables, 433
 - WHERE clause, 87–93
 - aggregate functions and, 178
 - comparison operators (<,<=,>,>=), 87
 - conditional operators (AND/OR/NOT), 87, 88
 - DELETE statement, importance in, 169
 - HAVING clause compared, 178
 - joining table conditions, 102
 - SQL92 SELECT syntax, 110
 - UPDATE statement, importance in, 166, 167
 - used in subquery, 186, 187
 - WHILE loops
 - execution control structures, 293
 - wildcard characters
 - PEAR, 462
 - using % in pattern matching, 91
 - using _ in pattern matching, 91
 - Windows
 - installing PostgreSQL on, 59–64
 - starting up psql, 74–75
 - using Windows installer, 59
 - WITH GRANT OPTION
 - GRANT command, 337
 - with-xyz options
 - configure script, 51
 - work_mem option
 - PostgreSQL.conf file, 314
 - working asynchronously
 - see* asynchronous working
 - working directory, changing, 119
 - Write Ahead Logging (WAL), 247
- X**
- x command (\x), psql, 121, 576
 - x option (-x)
 - psql, 119, 574
 - xml2sql.pl script, 488
 - X option (-X), psql, 114, 119, 574
 - XA resources
 - JDBC extension API, 491
 - XML
 - creating from SQL, 487–488
 - creating SQL from XML, 488–489
 - creating XML from DBI queries, 485–489
 - XML_RDB module, Perl, 485
 - xml2sql.pl script, 487
 - options, 488
 - XOR operator
 - Binary XOR operator, 271
- Y**
- YaST2 installation tool
 - installing PostgreSQL packages, 45
 - from SuSE packages, 46
 - Yellow Dog PPC
 - packages for download available at, 44
- Z**
- z command (\z), psql, 121, 576
 - Z option (-Z)
 - pg_dump utility, 340
 - z option (-z)
 - vacuumdb utility, 351
 - ZIP code
 - database design, 365