można tworzyć tabele, które eliminują duplikaty danych, są łatwiejsze w utrzymaniu i zapewniają większą elastyczność w pisaniu zapytań w celu uzyskania tylko żądanych danych.

Łączenie tabel z użyciem JOIN

Aby połączyć tabele w zapytaniu, używamy instrukcji `JOIN ... ON` (lub jednego z innych wariantów `JOIN`, które omówię w tym rozdziale). W zapytaniu instrukcja `JOIN` łączy jedną tabelę z inną istniejącą w bazie danych, używając dopasowanych wartości w kolumnach obu tabel. Składnia przyjmuje następującą postać:

```
SELECT *
FROM table_a JOIN table_b
ON table_a.key_column = table_b.foreign_key_column
```

Jest to podobne do podstawowej składni `SELECT`, której już się nauczyliśmy, ale zamiast używać jednej tabeli w klauzuli `FROM`, wstawiamy nazwę tabeli, podajemy słowo kluczowe `JOIN`, a następnie nazwę drugiej tabeli. Po słowach kluczowych `ON` określamy kolumny, których chcemy użyć do dopasowania wartości. Po uruchomieniu zapytania system sprawdza obie tabele, a następnie zwraca kolumny z obu tabel, w których wartości są zgodne w kolumnach użytych w klauzuli `ON`.

Złączenie oparte na równości między wartościami jest najczęściej stosowane w klauzuli `ON`, ale można też użyć dowolnego wyrażenia, które zwraca wartość *logiczną* `true` lub `false`. Na przykład możesz napisać takie złączenie, w którym wartości z jednej kolumny są większe lub równe wartościom w innej.

```
ON table_a.key_column >= table_b.foreign_key_column
```

To rzadkie, ale możliwe, jeśli Twoja analiza tego wymaga.

Łączenie tabel z użyciem kolumn kluczy

Rozważmy przykład łączenia tabel z użyciem kolumn kluczy: wyobraź sobie, że jesteś analitykiem danych, którego zadaniem jest sprawdzanie płac w poszczególnych działach agencji publicznej. Składasz wniosek dotyczący dostępu do informacji o wynagrodzeniach w agencji, oczekując prostego arkusza kalkulacyjnego z listą poszczególnych pracowników i ich wynagrodzeniem, ułożonymi w następujący sposób:

```
dept     location     first_name     last_name      salary
----     --------     ----------     ---------      ------
Tax      Atlanta      Nancy          Jones          62500
Tax      Atlanta      Lee            Smith          59300
IT       Boston       Soo            Nguyen         83000
IT       Boston       Janet          King           95000
```

Jednak nie dostajesz tego, o co Ci chodziło. W zamian agencja wysyła zrzut danych z systemu płac: tuzin plików CSV, z których każdy zawiera jedną tabelę z bazy. Czytasz dokument wyjaśniający układ danych (pamiętaj, aby zawsze o to poprosić!) i zaczynasz rozumieć znaczenie kolumn w każdej tabeli. Najbardziej istotne są dwie tabele: pierwsza nazwana `employees` i druga `departments`.

Używając kodu z listingu 6.1, stwórzmy wersje tych tabel, wstawmy wiersze i zbadajmy, jak połączyć dane w obu tabelach.

Korzystając z utworzonej na potrzeby tych ćwiczeń bazy danych `analysis`, wykonaj cały kod, a następnie przejrzyj dane, stosując podstawową instrukcję `SELECT` lub klikając nazwę tabeli w pgAdmin i wybierając **View/Edit Data ▶ All Rows**.

```
CREATE TABLE departments (
     dept_id bigserial,
     dept varchar(100),
     city varchar(100),
  ❶  CONSTRAINT dept_key PRIMARY KEY (dept_id),
  ❷  CONSTRAINT dept_city_unique UNIQUE (dept, city)
);

CREATE TABLE employees (
     emp_id bigserial,
     first_name varchar(100),
     last_name varchar(100),
     salary integer,
  ❸  dept_id integer REFERENCES departments (dept_id),
  ❹  CONSTRAINT emp_key PRIMARY KEY (emp_id),
  ❺  CONSTRAINT emp_dept_unique UNIQUE (emp_id, dept_id)
);

INSERT INTO departments (dept, city)
VALUES
     ('Tax', 'Atlanta'),
     ('IT', 'Boston');
INSERT INTO employees (first_name, last_name, salary, dept_id)
VALUES
     ('Nancy', 'Jones', 62500, 1),
     ('Lee', 'Smith', 59300, 1),
```