1) B-Tree Pages

Hai biến thể của b-tree được sử dụng bởi SQLite. "table b-tree" sử dụng khóa số nguyên có dấu 64 bit và lưu trữ tất cả dữ liệu trong các lá. “index b-tree” sử dụng các khóa tùy ý và không lưu trữ dữ liệu nào cả.

Một B-tree page là page nội thất hoặc page lá. Một page lá chứa các khóa và trong trường hợp của một bảng b-tree, mỗi khóa có dữ liệu liên đã được liên kết. Một page nội chứa K khóa cùng với K + 1 con trỏ đến các trang b-tree con. Một "con trỏ" trong trang b-tree bên trong chỉ là số trang số nguyên không dấu 32-bit của page con.

Số lượng khóa trên page b-tree nội, K, hầu như luôn luôn ít nhất là 2 và thường nhiều hơn 2. Ngoại lệ duy nhất là khi page 1 là page b-tree nội. Page 1 có ít hơn 100 byte không gian lưu trữ có sẵn, do sự hiện diện của tiêu đề cơ sở dữ liệu ở đầu trang đó và vì vậy, đôi khi (hiếm khi) nếu trang 1 là một page b-tree bên trong, nó có thể chỉ chứa một key đơn. Trong tất cả các trường hợp khác, K là 2 hoặc nhiều hơn. Giới hạn trên của K là nhiều key nhất có thể phù hợp trên page. Các key lớn trên index b-tree được chia thành các overflow page để không có key đơn nào sử dụng nhiều hơn một phần tư không gian lưu trữ có sẵn trên page và do đó mỗi page nội có thể lưu trữ ít nhất 4 khóa. Các khóa nguyên của table b-tree không bao giờ đủ lớn để yêu cầu tràn, vì vậy tràn khóa chỉ xảy ra trên index b-tree.

Định nghĩa độ sâu của một lá b-tree là 1 và độ sâu của bất kỳ lá b-tree nội nào lớn hơn một độ sâu tối đa của bất kỳ cây con nào của nó. Trong một cơ sở dữ liệu được hình thành tốt, tất cả các con của b-tree nội đều có cùng độ sâu.

Trong một page b-tree nội, các con trỏ và khóa luân phiên một cách hợp lý với một con trỏ ở cả hai đầu. (Câu trước cần được hiểu theo khái niệm - cách bố trí thực tế của các khóa và con trỏ trong page phức tạp hơn và sẽ được mô tả trong phần tiếp theo.) Tất cả các khóa trong cùng một page là duy nhất và được tổ chức hợp lý theo thứ tự tăng dần từ trái qua phải. (Một lần nữa, thứ tự này là rõ ràng, không phải vật lý. Vị trí thực tế của các khóa trong trang là tùy ý.) Đối với bất kỳ khóa X nào, con trỏ ở bên trái của X tham chiếu đến các page b-tree trên đó tất cả các khóa đều <= X. Con trỏ ở bên phải X tham chiếu đến các trang có tất cả các khóa > X.

Mỗi page b-tree có nhiều nhất một page b-tree cha. Page b-tree không có cha được gọi là trang gốc. Một page b-tree gốc cùng với sự đóng lại của các con của nó tạo thành một b-tree hoàn chỉnh. Có thể (và trên thực tế là khá phổ biến) để có một cây b hoàn chỉnh bao gồm một page duy nhất vừa là lá vừa là gốc. Bởi vì có các con trỏ từ cha đến con, mọi page của cây b-tree hoàn chỉnh đều có thể được định vị nếu chỉ biết trang gốc. Do đó, b-tree được xác định bằng số trang gốc của chúng.

Page b-tree hoặc là page table b-tree hoặc page index b-tree. Tất cả các page trong mỗi b-tree hoàn chỉnh đều thuộc cùng một loại: table hoặc index. Có một table b-tree trong file database cho mỗi bảng rowid trong lược đồ database, bao gồm các bảng hệ thống như sqlite\_schema. Có một index b-tree trong file database cho mỗi index trong lược đồ, bao gồm các index ngầm hiểu được tạo bởi các ràng buộc về tính duy nhất. Không có b-tree nào được liên kết với các bảng ảo. Triển khai bảng ảo cụ thể có thể sử dụng các shadow table để lưu trữ, nhưng các shadow table đó sẽ có các mục riêng biệt trong lược đồ database. Bảng WITHOUT ROWID sử dụng index b-tree thay vì table b-tree, do đó, có một index b-tree trong file database cho mỗi bảng WITHOUT ROWID. b-tree tương ứng với bảng sqlite\_schema luôn là table b-tree và luôn có page gốc là 1. Bảng sqlite\_schema chứa số trang gốc cho mọi bảng và chỉ mục khác trong file database.

Mỗi mục trong table b-tree bao gồm một khóa số nguyên có dấu 64 bit và tối đa 2147483647 byte dữ liệu tùy ý. (Khóa của table b-tree tương ứng với rowid của SQL table mà b-tree thực hiện.). Table b-tree nội chỉ chứa các khóa và con trỏ tới con. Tất cả dữ liệu được chứa trong các table b-tree lá.

Mỗi mục trong một index b-tree bao gồm một khóa tuỳ ý lên tới 2147483647 byte chiều dài và không có data.

Định nghĩa “payload” của một cell là phần độ dài tùy ý của cell. Đối với index b-tree, khóa luôn có độ dài tùy ý và hiển nhiên “payload” là khóa. không có phần tử độ dài tùy ý trong các cell của các page tabel b-tree nội nên các cell đó không có “payload”. Các page lá table b-tree chứa nội dung độ dài tùy ý nên các cell trên các page đó, “payload” là nội dung.

Khi kích thước của payload của cell vượt quá một ngưỡng nhất định, chi một ít byte đầu tiên của payload được lưu trữ trên b-tree page và số còn lại được lưu trong linked-list của nội dung overflow page.

