ALS1 – Přednáška 5

1 R-stromy

R-strom je výškově vyvážený strom pro správu prostorových dat, který je strukturou podobný B-stromu. Poprvé byly představeny v [1], informace v tomto dokumentu jsou čerpány z [1,2].

1.1 Definice

Nechť M je maximální počet záznamů, které mohou být v jednou uzlu, a $m \leq \frac{M}{2}$ je minimální počet záznamů v uzlu. R-tree splňuje následující vlastnosti:

- Každý uzel obsahuje m až M indexových záznamů, pokud to není kořen.
- Pro každý indexový záznam (I, id) v listovém uzlu, I je nejmenší obdélník obsahující n-dimenzionální datový objekt reprezentovaný daným id.
- Každý nelistový uzel, vyjma kořene, má m až M potomků.
- Každý záznam (I, c) v nelistovém uzlu, I je nejmenší obdélník, který obsahuje všechny obdélníky v potomku c.
- Kořen má alespoň dva potomky, pokud to není list.
- Všechny listy mají stejnou hloubku.

Výška R-stromu obsahující N indexových záznamů je nejvýše $|\log_m N| - 1$, protože počet větví u každého každého uzlu je nejméně m. Maximální počet uzlů je $\left|\frac{N}{m}\right| + \left|\frac{N}{m^2}\right| + \cdots + 1$. Nejhorší prostorové využití R-stromu je $\frac{m}{M}$. Uzly budou mít spíše větší počet záznamů, což zlepší prostorové využití. Pokud uzly mají více než 3–4 záznamy, strom je velmi široký a téměř všechen prostor je použitý na listové uzly obsahující indexové záznamy. Parametr m může být měněn jako součást zlepšování výkonu.

Obrázek 1 ukazuje příklad R-stromu a rozložení jeho obdélníků.

1.2 Vyhledávání

Vyhledávání v R-stromu sestupuje od kořene dolů – podobně jako vyhledávání v B-stromu. Na rozdíl od B-stromu ale může být potřeba vyhledávat ve více než jednom podstromu navštíveného uzlu. Takže není možné zaručit dobrý výkon v nejhorším případě. Přesto u většiny druhů dat procedury R-Tree-Insert a Delete udržují R-strom v podobě, která umožňuje vyhledávacímu algoritmu eliminovat irelevantní části indexovaného prostoru a zkoušet jen data blízko oblasti vyhledávání.

V následujícím popisu je označen obdélník indexového záznamu E písmeny I_E a id či c jsou označeny p_E. Symbol $\lozenge$ představuje překryv.

Algoritmus R-Tree-Search: Bere jako argumenty R-strom s kořenem T, vyhledávaný obdélník S; najde všechny indexové záznamy, které překrývají S.

S1 [Prohledej podstromy] Pokud T není list, prozkoumej všechny záznamy E a zjisti jestli $S \lozenge I_E$. Pro všechny takové E vyvolej R-Tree-Search se stromem, který je zakořeněný v p_E.

S2 [Prohledej list] Pokud T je list, projdi všechny záznamy E a zjisti jestli $S \lozenge I_E$. Pokud ano, E je výsledný záznam.

1.3 Vkládání

Vkládání indexových záznamů pro nová data do R-stromu je podobné vkládání do B-stromu v tom, že nové indexové záznamy jsou přidávané do listů, uzly které jsou přeplněné se rozlomí a rozlamování uzlu se šíří stromem směrem nahoru.

Algoritmus R-Tree-Insert – vkládá nový indexový záznam E do stromu T.

I1 [Najdi pozici pro nový záznam] Vyvolej proceduru R-Tree-Choose-Leaf na výběr uzlu L, do kterého bude umístěn E.

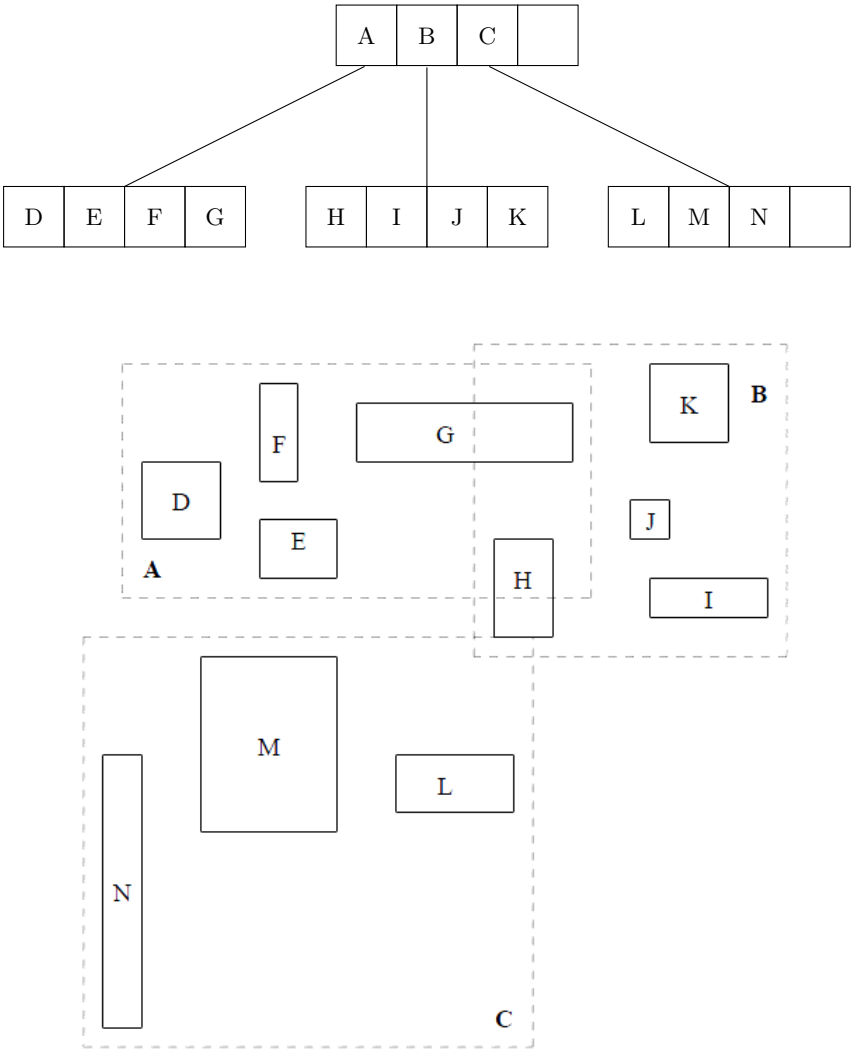
I2 [Přidej záznam do listového uzlu] Pokud L má místo pro další záznam, vlož tam E. Jinak vyvolej R-Tree-Split-Node a získej L a LL obsahující E a všechny staré záznamy z L.

I3 [Rozšiř změny nahoru] Vyvolej proceduru R-Tree-Adjust-Tree na L (i s LL, pokud bylo provedeno rozlomení).

I4 [Zvyš výšku stromu] Pokud se rozlamování uzlu postupně dostalo až ke kořeni, vytvoř nový kořen, jehož potomci budou výsledné dva uzly (L a LL).

Algoritmus R-Tree-Choose-Leaf: Vyber list, do kterého se umístí nový indexový záznam E.

CL1 [Inicializace] Nastav $N \leftarrow$ kořen.



Obrázek 1. R-strom a rozhložení jeho obdélníků

CL2 [Kontrola na list] Pokud N je list, vrať N.
CL3 [Vyber podstrom] Pokud N není list, najdi záznam F v N tak, že I_F potřebuje nejmenší rozšíření, aby $I_E \subseteq I_F$. „Remízy“ vyřeš výběrem záznamu s menším obdélníkem.
CL4 [Sestupuj k až listu] Nastav $N \leftarrow p_F$ a opakuj od kroku CL2.

Algoritmus R-Tree-Adjust-Tree: Vystupuj od listu ke kořenu, upravuj velikosti obdélníků,

AT1 [Inicializace] Nastav $N \leftarrow L$; pokud L byl rozlomen, nastav $NN \leftarrow LL$ na výsledný druhý uzel.
AT2 [Kontrola na dokončení] Pokud N je kořen, skonči.
AT3 [Uprav obdélník v předkovi] Nechť P je předek uzlu N a E_N je záznam v P obsahující N. Uprav I_{E_N} tak, aby těsně obsahovalo obdélníky všech záznamů v N.
AT4 [Šiř rozlamování uzlů směrem ke kořenu] Pokud máme NN z předchozího prolamování, vytvoř pro něj nový záznam E_{NN} s $p_{E_{NN}} \leftarrow NN$ a $I_{E_{NN}} \leftarrow$ uzavírající všechny obdélníky v NN. Přidej E_{NN} do P pokud je tam místo. V opačném případě vyvolej R-Tree-Split-Node, aby se vytvořilo P a PP obsahující E_{NN} a všechny staré záznamy z P.
AT5 Nastav $N \leftarrow P$; $NN \leftarrow PP$, pokud nastalo rozlomení, a opakuj od kroku AT2.

Alogismus R-Tree-Split-Node bude popsán později.

1.4 Mazání

Algoritmus Delete. Odstraní indexový záznam E z R-stromu.

D1 [Najdi uzel, který obsahuje E] Vyvolej R-Tree-Find-Leaf a najdi tak listový uzel L obsahující E. Pokud takový uzel neexistuje, skonči.
D2 [Smaž záznam] Odstraň E z L.
D3 [Šiř změnu] Vyvolej R-Tree-Condense a předej jí L.
D4 [Zmenši výšku stromu] Pokud kořen má jen jednoho potomka po předchozím kroku, udělej z toho potomka nový kořen.

Algoritmus R-Tree-Find-Leaf

FL1 [Prohledej podstromy] Pokud T není list, najdi záznamy F v T, tak že $I_E \subseteq I_F$. Pro každý takový záznam vyvolej R-Tree-Find-Leaf pro strom, který je zakořeněný v p_F dokud záznam E není nalezen, nebo nebyly prohledány všechny záznamy.
FL2 [Prohledej list] Pokud T je list, najdi v něm záznam shodný s E. Pokud je takové E nalezeno, vrať T.

Algoritmus R-Tree-Condense. Pro listový uzel L ze kterého byl odstraněn záznam: pokud má málo záznamů, odstraň uzel, a přesuň jeho záznamy. Šiř změnu, pokud je to potřeba. Zmenšuj všechny obdélníky na cestě ke kořeni, pokud to jde.

CT1 [Inicializace] Nastav $N \leftarrow L$, $Q \leftarrow \emptyset$; Q je množina eliminovaných uzlů.
CT2 [Najdi předka] Pokud N je kořen, pokračuj bodem 1.4. Jinak nechť P je předka N, E_N záznam v P, ve kterém je N.
CT3 [Odstraň nedostatečně naplněný uzel] Pokud N má méně než m záznamů, smaž E_N z P a přidej N do Q.
CT4 [Uprav obdélník] Pokud N nebyl eliminován, uprav I_{E_N} tak, aby těsně obsahovala všechny záznamy v N.
CT5 [Postup nahoru] Nastav $N \leftarrow P$ a pokračuj bodem 1.4.
CT6 [Znovu vlož osiřelé záznamy] Znovu-vlož všechny záznamy uzlů v množině Q. Záznamy z listových uzlů jsou vloženy tak jak je popsáno v Algoritmu R-Tree-Insert, ale záznamy z vnitřních uzlů musí být uloženy výše ve stromu, tak že listy jejich závislých podstromů budou na stejné úrovni, jako listy hlavního stromu.

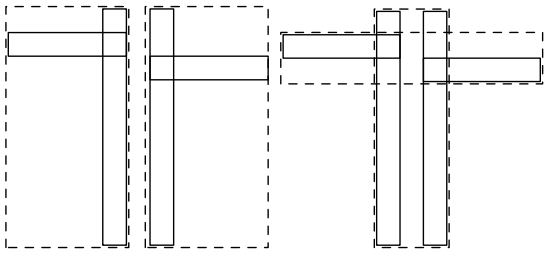
1.5 Rozlomení uzlu

Kvůli uložení nového záznamu do plného uzlu, který obsahuje M záznamů, je nutné rozdělit kolekci $M + 1$ záznamů mezi dva uzly. Toto rozdělení by mělo být provedeno tak, aby bylo nepravděpodobné, že v následných použití Algoritmu R-Tree-Search se neprohledávaly oba nové uzly. Protože rozhodnutí, jestli uzel bude navštíven závisí na na tom, jestli jeho obdélník zasahuje hledanou oblast, měla by být oblast dvou nových obdélníků co nejmenší. Viz obr. 1.5.

Stejné kritérium se používalo v proceduře R-Tree-Choose-Leafk rozhodnutí kam se vloží nový idexový záznam. Na každé úrovni ve stromu byl vybírán ten podstrom, jehož obdélník by se zvětšil nejméně.

Nyní uvedu několik algoritmů pro rozklad množiny $M + 1$ záznamů do dvou tříd – každá pro jeden nový uzel.

Bruteforce Vyzkouší všechny možnosti rozdělení $M + 1$ záznamů do 2 skupin o minimálně m uzlech. Vybere tu možnost, která bude mít nejmenší obsah pokrývajících obdélníků.



Obrázek 2. Špatný split (vlevo) a dobrý split (vpravo)

Kvadratický split Tento algoritmus se pokouší najít split na malé oblastí, ale není zaručeno, že najde ty nejmenší. Složitost je kvadratická vzhledem k M a lineární vzhledem k počtu dimenzí. Algoritmus vybere dva záznamy z celkových $M + 1$ záznamů, jako první prvky nových skupin tak, že vybere tu dvojici prvků, které by nejvíc plýtvaly místem, kdyby byly vloženy do jedné skupiny. Plýtvané místo je obsah oblasti pokrývající oba záznamy mínus obsah oblastí samotných záznamů. Zbývající záznamy jsou přidělovány skupinám po jednom. V každém kroku je pro každý záznam vypočítán obsah o který by musela byt současná oblast skupiny rozšířena. Vybrán je ten záznam u kterého je největší rozdíl ve velikosti té oblasti.

Algoritmu SPLIT: Rozděl $M + 1$ záznamů do dvou skupin

QS1 [Vyber první záznam pro každou třídu] Vyvolej proceduru PICKSEEDS na výběr dvou záznamů, které se stanou prvními prvky tříd rozkladu. Přiřaď každý jedné třídě.

QS2 [Kontrola na konec] Pokud všechny záznamy byly přiřazeny, skonči. Pokud jedna z tříd má tak málo záznamů, že všechny nepřiřazené záznamy se musí přidat do této třídy, aby měla alespoň m záznamů, proveď toto přiřazení a skonči.

QS3 [Vyber záznam k přiřazení] Vyvolej proceduru PICKNEXT k výběru dalšího záznamu, který bude přiřazen. Přiřaď ho do té třídy, jejíž pokrývající obdélník bude muset být rozšířen méně, aby zahrnul jako obdélník. „Remízu“ vyřeš výběrem třídy s menším pokrývajícím obdélníkem, pak ten s menším počtem záznamů. Pokračuj krokem 1.5

Algoritmus PICKSEEDS: Vyber dva záznamy, aby byly prvními prvky skupin.

PS1 [Vypočítej neefektivnost přiřazení záznamů jedné třídě] Pro každý pár záznamů E_1 a E_2, vytvoř obdélník, který pokrývá I_{E_1} a I_{E_2}, vypočítej $d = \text{obsah}(J) - \text{obsah}(I_{E_1}) - \text{obsah}(I_{E_12})$.

PS2 [Vyber nejméně efektivní pár] Vyber pár s největším d.

Algoritmus PICKNEXT

PN1 [Vypočítej cenu vložení každého záznamu do každé třídy] Pro každý záznam E, který ještě není přiřazený žádné třídě, vypočítej $d_1 =$ obsah, o který se zvětší pokrývající obdélník první třídy, podobně vypočítej d_2 pro druhou třídu.

PN2 [Vyber záznam s nejvyšší preferencí] Vyber záznam s maximálním rozdílem d_1 a d_2.

Lineární split Tento algoritmus je lineární vzhledem k M a vzhledem k počtu dimenzí. Lineární split je identický s kvadratickým splitem, jen používá jinou verzi PICKSEEDS. PICKNEXT jednoduše vybírá kterýkoli ze zbývajících záznamů.

Algoritmus PICKSEEDS: Vyber dva záznamy, aby byly prvními prvky skupin.

LS1 [Vyber extrémní obdélníky mezi všemi rozměry] Najdi obdélník s nejpravějším levým okrajem a obdélník s nejlevějším pravým okrajem. To udělej pro všechny dimenze (tedy pro vertikální: Najdi obdélník s nejhornějším dolním okrajem a obdélník s nejspodnějším horním okrajem, atd.) Zapamatuj si tyto páry a jejich separaci (tj. rozdíl mezi nejpravějším levým okrajem a obdélník s nejlevějším pravým okrajem, etc.).

LS2 [Uprav] Normalizuj separaci vydělením odpovídajícím rozměrem obdélník pokrývající všechny záznamy.

LS3 [Vyber nejextrémnější pár] Vyber pár s největší normalizovanou separací podle kterékoli dimenze.

1.6 Varianty R-stromů

Dynamické R-stromy: R+stromy, R*-stromy, Hilbertovy R-stromy, kompaktní R-stromy, cR-stromy
Statické R-stromy: Packed R-tree, R*-stromy, Hilbert Packed R-tree, STR R-tree, etc.

Reference

1. A. Guttman, R-Trees: A dynamic index structure for spatial searching, in Proc. of ACM SIGMOD, June 1984, pp. 47–57.
2. Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis. *R-trees: Theory and Applications.* Series in Advanced Information and Knowledge Processing. Springer, 2005.