**Relation DB**

Այն բազաները որոնցում տվյալները ներկայացվում են աղյուսակենրի տեսքով որոնց միջև կարելի է սահմանել կապեր կոչվում են ռելացիոն DB – ներ:

DB – ի հետ աշխատելու համար կան հետևյալ տարբերակները՝

* JDBC API ամենացածր մակարդակը, որտեղ DB query – ները կատարվում են ձեռքով:
* Jdbc Template – Spring Framework – ի կազմում, աշխատանքնի մեջ մտցնում է աբստրակցիա, որոշ գործողություններ կատարում է տակից:
* Hibernate – ամենաբարձր մակարդակը գրեթե չեն օգտագործվում ձեռքով գրված query – ներ: Ավտոմատ convert է անում օբյեկտները աղյուսակների տողի և հակառակը, որը կոչվում է ORM (Object relational Mapping):

ProsgreSQL – ում ունիկալ id ավտոմատ գեներացնելու համար օգտագործվում է sequence օբյեկտի միջոցով, որոնք ստեղծվելուց հետո պահվում են առանձին folder -ում և տալիս են ամբողջ թվերի հերթականությամբ՝

CREATE SEQUENCE some\_sequence

Ստուգելու համար՝ select nextval(‘some\_sequence’)

Field – ը ավտոինկրեմենտով դարձնելու համար պետք է դրա տիպը նշվի SERIAL, այս դեպքում արժեքը առաջվա նման int կլինի բայց postgres – ը տակից կստեղծի նաև sequence հետևյալ անունով`

TableName\_Field\_Sequence

Կա սինտաքսի նոր տարբերակ SERIAL – ի փոխարեն՝

id int GENERATED BY DEFAULT AS IDENTITY

այս դեպքում նորից կստեղծվի sequence

նոր սինտաքսիսի պատճառը համապատասխանում է SQL ստանդարտներին և նման է այլ DB -ների սինտաքսին

SQL սահմանափակումներ Constraints – ը կիրառվում է field- երի թույլատրելի արժեքների վալիդացիաների համար:

NOT NULL – չի կարող լինել null

UNIQUE – միայն ունիկալ արժեք

PRIMARY KEY – իր մեջ ներառում է UNIQUE + NOT NULL

FOREIGN KEY – user\_id int REFERENCES Parent\_Table (id)

CHECK – կարող են նշվել այլ սահմանափակումներ

create tabel Person (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY<

name varchar NOT NULL,

age int check (age > 0),

email varchar UNIQUE

)

Parent աղյուսակը կարող է գյություն ունենալ ինքնությույն իսկ child աղյուսակը իր մեջ պարունակում է paren – ին:

DB – ում parent և child աղյուսակների կապերը կարող են լինե

* One to one – մեկ քաղաքացին ունի մեկ անձնագիր, մեկ դպրոցը ունի մեկ տնօրեն, մեկ երկիրը մեկ նախագահ: Այս դեպքում child – ում PRIMARY KEY – ը հավասար է PRIMARY KEY – ին, այսինքն հնարավորություն չի լինի parent – ի տողը մեկից ավելի անգամ կապել child – ի տողի հետ քանի որ PRIMARY KEY – ը իր մեջ ներառում է նաև UNIQUE սահմանափակումը:
* One to many – մեկ գնորդը կարող է ունենալ շատ պատվերներ, մեկ գրադարանը շատ գրքեր, մեկ ուսուցիչը շատ աշակարտներ: Ամենահաճախ հանդիպող կապն է որի ժամանակ child աղյուսակը որպես FOREIGN KEY իր մեջ պահում է parent աղյուսակի PRIMARY KEY - ը
* Many to many – մեկ դերասանը կարող է նկարահանվել շատ ֆիլմերում և մեկ ֆիլմում կարող են նկարահանվել շատ դերասաններ, ուսանողը կարող է մասնակցել շատ դասերի մեկ դասին կարող են մասնակցել շատ ուսանողներ: Այս դեպքում աղյուսակները իրենց մեջ չեն պահում foreign key – եր և դրա փոխարեն ստեղծվում է 3 – րդ աղյուսակը որը կոչվում է join table որը արդեն չունի PRIMARY KEY և որի մեջ պահվում են միայն աղյուսակների FOREIGN KEY – երը (աղյուսակը ստեղծելիս id դաշտերը հայտարարվում են foreign key – ի սինտաքսով REFERENCES – Ի միջոցով): Նման աղյուսակներում նույն տողը չի կարող կրկնվել: Այսինքն ապահովվում է ոչ թե մեկ սյունյակի այլ բոլոր սյունյակների իրար հետ միասին UNIQUE լինելը:

Աղյուսակների միավորման համար օգտագործվում են join – ներ, որոնց արդյունքում ստացված միավորված աղյուսակը գեներացվում է մեկ անգամ և չի պահվում DB – ում: Join - ները կարող են լինել՝

* Inner Join – 2 աղյուսակների հատումն է, այսինքն հատված աղյոսակում չեն ներառվի Parent աղյուսակի այն տողերը որոնց համընկնումներ չեն եղել

SELECT \* FROM ParentTable JOIN ChildTable ON ParentTable.field\_Id = CHILDTABLE.field\_id

* Outer Join (Left Join, Right Join) – 2 աղյուսակների միավորումն է որի ժամանակ վերջնական աղյուսակի մեջ ներառվում են հավասարման ձախ / աջ մասի նաև բոլոր տողերը՝ բացակայող զույգերի համար null արժեքներով

SELECT \* FROM ParentTable LEFT JOIN ON ParentTable.field\_Id = CHILDTABLE.field\_id

SELECT \* FROM ParentTable RIGHT JOIN ON ParentTable.field\_Id = CHILDTABLE.field\_id

Right Join – ը չի օգտագործվում քանի որ այն կարելի է արտահայտել Left Join – ի միջոցով, աղյուսակների տեղեր ը query – ում փոխելու միջոցով՝

ParentTable RJ ChildTable = ChildTable LJ ParentTable

* Cross Join – ի դեպքում ձախ կողմի բոլոր տողերը համադրվում են աջի բոլոր տողերի հետ

SELECT \* FROM ParentTable CROSS JOIN ChildTable

**Indexing**

Query – ների ավելի արագ իրականացման համար կարող են օգտագործվել DB ինդեքսներ, որպեսզի select – ի ժամանակ բոլոր տողերի վրայով անցնելու կարիք չլինի: Կոնկրետ սյունյակի համար ինդեքսսավորում օգտագործելիս ստեղծվում է առանձին աղյուսակ որն իր մեջ պարունակում է ինդեսավորված և id սյունյակները ընդ որում արժեքները դասավորվում ինդեքսավորված սյունյակի աճման կարգով, որը հնարավորություն է տալիս DB – ին օգտագործել ավելի արագ ալգորիթմներ(օր binar search) select – ի ժամանակ: Օր 5 մլն տողի համար սովորական search – ի համար կպահանջվի միջինը 2.5 մլն իտերացիա, իսկ binary search – ի դեպքում log2(5մլն) = 23 իտերացիա:

Աղյուսակի սյունյակի ինդեքսավորման բացասական կողմերից է նոր տողի ավելացումը, որովհետև այս դեպքում արժեքները սորտավորված են և պետք է գտնել նոր արժեքին համապատասխան տեղը այն insert անելու համար: Ինդեքսները զբաղեցնում են լրացուցիչ հիշողություն:

Այսինքն ինդեքսներ նպատակահարմար է ստեղծել միայն այն սյուների համար որոնցում search կատարվում է հաճախ:

PostgreSQL – ում primary key – ը ինդեքսավորվում է default:

Ինդեքսավորման համար՝

CREATE INDEX ON TABLE\_NAME(column\_name) USING <indexing type>

Ինդեքսավորման տիպերը նույնպես տարբեր են որոնք կատարում են սորտավորում տարբեր ալգորիթմներով՝

* B-Tree Index
* Hash Index
* Gin & Gist Index // տեքստային սյուների համար
* BRIN Index

Ինդեքսները շատ լայն թեմա է, բայց ընդհանրացված կարելի է ասել որ ինդեքսները օգտագործվում են սյուները սորտավորելու համար դրանցում search – ը ավելի արագ դարձնելու նպատակով:

**DB Cascading**

Foreign key – ի կոնտեքստում կարևոր է DB – ի կասկադավորումը: Կասկադավորման դեպքում իրար հետ կապված աղյուսակներից Parent աղյուսակից որևէ տողի ջնջվելուց հետո child աղյուսակում ջնջվում է նաև այն տողը որը foreign key – ով կապված է Parent - ի ջնջված տողի հետ:

Առանց կասկադավորման ջնջելու դեպքում կունենանք error:

Կասկադավորումը տեղի է ունենում table – ի ստեղծման ժամանակ երբ հայտարարվում է foreign key, օր՝

User\_id int REFERENCES Parent\_Table(id) ON DELETE <cascade type>

Տիպերը կարող են լինել՝

CASCADE – ջնջում է տողը child աղյուսակում parent – ի տողը ջնջվելու դեպքում

SET NULL – parent – ի տողը ջնջվելուց հետո child – ում դրա foreign key - ին վերագրվում է null

RESTRICT – default վարք այսինքն ջնջելուց կունենանք error