Universitatea „Alexandru Ioan Cuza” din Iaşi
Facultatea de Economie şi Administrarea Afacerilor
Informatică Economică

Comparații între prelucrarea datelor în SQL şi pachetul ”Tidyverse” din R

Coordonator,
Prof. Univ. Dr. Fotache Marin

Absolvent,
Irimia Mihaela

2019
IAŞI

Cuprins

Lista figurilor

Lista tabelelor

Introducere

Mediul de afaceri în care activează companiile în prezent este semnificativ mai complex decât în deceniile precedente, aflându-se, de altfel, într-o permanentă schimbare. Fie că vorbim de cele ce activează în mediul privat, fie că vorbim de cele ce activează în sectorul public, organizațiile sunt forțate să se adapteze frecvent unor condiții noi, și să răspundă într-un timp cât mai scurt cerințelor fluctuante. Abilitatea acestora de integrare și analiză a datelor provenite din surse variate devine, astfel, un factor determinant în ceea ce privește succesul lor.

La ora actuală, se consideră peste 90% din datele existente la nivel mondial au fost generate în ultima perioadă, iar aproximativ 90% dintre acestea sunt date nestructurate[1]. Acest lucru a determinat organizațiile să acorde o importanță deosebită datelor și să investească sume uriașe în instrumente de colectare, stocare și procesare a acestora. Prelucrarea și analizarea unor volume imense de date au constituit premisa unei mai bune înțelegeri a trecutului și a unei anticipări mai riguroase a condițiilor viitoare[2], ceea ce a oferit noi oportunități pentru descoperirea și crearea de valoare.

Creșterea accelerată a volumelor de date colectate a determinat dezvoltarea de noi tehnologii, care să permită stocarea cât mai eficientă a acestora, dar și care să proceseze și să extragă informațiile dorite într-un timp cât mai scurt în vederea utilizării acestora în procesul de luare a deciziilor. Nevoia de informații de înaltă calitate în demersurile decizionale, alături de dezvoltarea de software-uri din ce în ce mai performante au determinat apariția sistemelor de tip Business Intelligence and Analytics (BI&A).

Lucrarea de față va sintetiza principalele caracteristici și capabilități ale sistemelor de tip BI&A, în vederea unei examinări a provocărilor și oportunităților asociate acestora. De asemenea, vor fi analizate caracteristicile limbajelor de procesare a datelor R, respectiv SQL, a căror utilizare în cadrul sistemelor de tip BI&A facilitează accesarea și prelucrarea volumelor mari de date într-o maniera interactivă și ad-hoc[3]. Vor fi urmărite avantajele și dezavantajele utilizării lor și modul în care pot fi folosite pentru a aduce un plus unei organizații.

Cuvinte cheie: *baze de date, RStudio, SQL, limbajul R, interogări, limbaje de programare*

[1]Gang-Hoon, K., Trimi, S., Ji-Hyong, C., Big Data Applications in the Government Sector: A Comparative Analysis among Leading Countries, Communications of the ACM, vol. 57, nr. 3, 2014.
[2]Mazareanu, V., The Intelligence in Business Intelligence, Analele Științifice ale Universității "Alexandru Ioan Cuza", Științe Economice, Iași, 2006.
[3] Fotache, M., *Data Processing Languages for Business Intelligence. SQL vs. R*, Informatica Economică vol. 20, nr. 1, 2016.

Capitolul I Limbaje de procesare a datelor utilizate în BI&A

1.1. Despre sistemul de tip Business Intelligence and Analytics (BI&A)

Din ce în ce mai multe organizații apelează la sisteme de tip BI&A, pe de o parte pentru a înțelege modul în care funcționează afacerea, iar pe de altă parte de a afla de ce se află în punctul în care se află și cum ar putea să-și îmbunătățească performanța afacerii.

Soluțiile de tip BI&A oferă managerilor posibilitatea de a analiza activitatea desfășurată în cadrul organizației (afacerii) din perspective diferite și noi. Prin aplicarea modelelor analitice, identificarea informațiilor importante, ascunse în volumul mare de date, necesară pentru sprijinirea procesului decizional, devine mai ușoară.

Apariția sistemelor de tip BI&A a fost determinată de creșterea exponențială a volumului de date colectat de o organizație și diversificarea surselor de informații ale acesteia. Prin urmare, soluțiile de tip BI&A pot fi integrate cu sistemele operaționale existente în cadrul unei organizații, ceea ce face ca datele ce urmează a fi analizate să fie mai ușor de preluat și prelucrat.

1.2. Definirea conceptului de business intelligence and analytics

Nu există o singură definiție dată conceptului de Business Intelligence (B&IA) unanim acceptată în literatura de specialitate. Constatăm, de asemenea, că definițiile date acestui termen variază de la o sursă la alta, însă toate au același scop comun acela de a îmbunătăți și sprijini procesul decizional.

Termenul de "**Business Intelligence & Anaytics**" reprezintă un concept ce grupează sub aceeași umbrelă instrumente atât din domeniul informaticii cât și din domeniul afacerii, instrumente ce sunt utilizate pentru a transforma datele colectate în cadrul organizației în informații, a informațiilor în decizii, respectiv a deciziilor în acțiuni sau planuri de acțiune[4, 5].

Acest concept nu presupune altceva decât utilizarea tuturor datelor, provenite atât din surse interne cât și externe, de care dispune o organizație, pentru a sprijini și îmbunătăți procesul decizional.

[4]Coardos, D., Marinescu, I., A., Soluții de tip BI pentru asistarea deciziilor în administrarea publica locala, Revista Română de Informatică și Automatică, vol. 25, nr. 2, 2015.
[5]Eckerson, W., Smart Companies in the 21st Century: The Secrets of Creating Successful Business Intelligence Solutions, The Data Warehousing Institute, Seattle, 2003, p. 5.

Prin urmare, sistemele de tip BI&A pot fi privite ca fiind un set de procese și tehnologii ce convertesc datele în informații prețioase și utile respectiv planuri de acțiune[6], cu scopul de a îmbunătăți și sprijini procesul decizional.

Obiectivul sistemelor de tip BI&A este de a permite accesul la toate datele deținute de organizație în mod interactiv, de a permite manipularea și procesarea datelor, respectiv de a permite analiștilor și managerilor să realizeze analize complexe. Prin urmare, analiza datelor istorice și a celor curente permite obținerea unor informații prețioase pe baza cărora manageri pot lua decizii mai bune și mai eficiente.

1.3. Evoluția sistemelor de tip BI&A

Văzute ca fiind o "reîncarnare" a sistemelor suport a deciziilor, și câteodată referite în literatura de specialitate ca sisteme de tip BI&A[7], BI a devenit un termen popular în domeniul afacerilor și comunităților IT abia în anul 1990. La finalul anilor 2000, sistemele de tip BI integrează o nouă componentă cheie Business Analytics (Analiza afacerii)[8] și cuprinde o categorie largă de aplicații, tehnologii și procese pentru colectarea, stocarea, accesarea și analizarea datelor, având ca scop principal îmbunătățirea și sprijinirea procesului decizional[9].

În lucrarea Business Intelligence and Analytics: from big data to big impact, Chen și alți utilizează BI&A ca un termen unificat și tratează analiza big data ca fiind o nouă direcție de cercetare a BI&A. De asemenea, în cadrul lucrării sunt definite trei etape de evoluție: BI&A 1.0, BI&A 2.0 și BI&A 3.0. În ceea ce urmează vom face o scurtă prezentare a celor trei generații de sisteme de tip BI&A[10]:

BI&A 1.0

Sistemele BI&A din generația 1.0 au apărut în ani '90 și sunt orientate pe date structurate, colectate și stocate cu ajutorul sistemelor de gestiune a bazelor de date relaționale. Tehnicile de analiză a datelor utilizate în astfel de sisteme sunt cele bazate pe metode statistice și cele de tip data mining.

---

[6]Chaudhuri, S, Dayal, U., Narasayya,V., An Overview of Business Intelligence Technology, Communications of the ACM, Vol. 54 No. 8, p. 88-98, 2011.

[7]Fotache, M., Data Processing Languages for Business Intelligence. SQL vs. R, Informatica Economică vol. 20, nr. 1, 2016.

[8]Chen, H., Chiang, R., H., L., Storey, V., C., Business Intelligence and Analytics: from big data to big impact, MIS Quarterly, Vol. 36, nr. 4, pp. 1165-1188, 2012.

[9]Muntean, M., Surcel, T., Agile BI – The Future of BI, Informatica Economică vol. 17, nr. 3, 2013.

[10]Mohammed, J., Business Intelligence and Analytics Evolution, Applications, and Emerging Research Areas, International Journal of Engineering Science and Innovative Technology (IJESIT) Vol. 4, 2, 2015.

BI&A 2.0

Sistemele de tip BI&A din generația 2.0 sunt orientate pe text și pe analiza conținutului web. Permit analizarea datelor nestructurate provenite de la surse din domeniul social sau economic: conținutul documentelor web, email, mesaje tweitter, postări facebook, video și alte tipuri de conținut web. Aceste surse reprezintă o adevărată mină de aur pentru organizații deoarece permite înțelegerea nevoilor consumatorilor și descoperirea de noi oportunități de afaceri[11].

BI&A 3.0

Apariția și dezvoltarea BI&A 3.0 a fost determinată de numărul mare de telefoane mobile și tablete conectate la internet. În prezent, miliarde de senzori sunt conectați prin intermediul dispozitivelor mobile la internet. Provocare constă în colectarea datelor nestructurate provenite de la acești senzori și creare unor sisteme capabile să analizeze și să valorifice aceste date. BI&A 3.0 este orientat pe colectarea și analizarea datelor provenite de la aplicațiile mobile și senzori.

În ultimi ani, sistemele de tip BI&A au apărut ca un domeniu de cercetare a sistemelor suport a deciziilor, bucurându-se de un interes sporit din partea oamenilor de știință, dar și din partea organizațiilor, deoarece sistemele de tip BI&A pot contribui semnificativ la îmbunătățirea procesului decizional și implicit la îmbunătățirea performanțelor organizaționale.

1.4. Componentele sistemelor de tip BI

Activitatea desfășurată de o organizație presupune colectarea unui volum mare de date format din diferite tipuri de date și întreținerea unui număr mare de surse de date de diferite dimensiuni, eterogene. Prin urmare, principalul obiectiv al acestor organizații este integrarea acestor date cu scopul de a obține un acces eficient și o imagine de ansamblu asupra acestora.

Sistemele de tip BI cuprind trei zone importante: ***zona de achiziție a datelor sau zona de back-end*** - care extrage datele din sistemele operaționale ale organizației și surse externe și le încarcă în depozitul de date, ***zona depozitului de date*** - care conține datele și aplicațiile asociate**,** ***zona de analiza sau zona de front-end*** - ce permite utilizatorilor să acceseze și să analizeze datele stocate în depozitul de date.

[11]Antoniou, G., Papoglou, N., Business Intelligence & Analytics (BI&A) systems: measuring End-User Computing Satisfaction (EUCS), disertatie, 2015, p. 8, disponibila la http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=7365779&fileOId=7365851, accesat la 05.03.2019.

1.4.1. Zona de achiziție a datelor

La nivelul unei întreprinderi avem de-a face cu volume foarte mari de date, care provin din sursele operaționale ale întreprinderii cunoscute și sub numele de surse interne și/sau din surse externe. Sursele interne de date sunt reprezentate de sistemul operațional și/sau alte fișiere, în timp ce sursele externe de date sunt reprezentate de clienți, parteneri, piață etc. Este important de menționat faptul că, de obicei, volumul datelor provenite din surse interne este mult mai mare comparativ cu cel al datelor provenite din surse externe. Prin urmare, integrarea acestor date provenite din surse diferite reprezintă principala problemă cu care se confruntă organizațiile.

Progresele înregistrate în domeniul tehnologiei informației au condus la extinderea sistemelor de procesare a datelor de la aplicații autonome la sisteme complexe de tip BI&A în care, informațiile provenite din surse multiple, eterogene sunt integrate într-un singur loc - într-un depozit de date - cu scopul de a fi accesate, combinate, procesate și analizate în funcție de fiecare aspect al afacerii.

Tehnologiile utilizate pentru pregătirea și încărcarea datelor necesare sistemul de tip BI&A sunt referite în literatura de specialitate ca instrumente ETL (Extract-Transform-Load). Instrumentele back-end asociate depozitelor de date au rolul de a popula și a reîmprospăta datele conținute de acesta[12]. Tehnologiile de extragere, transformare și integrare au rolul de a extrage și a reorganiza datele provenite din surse multiplele.

Calitatea datelor importate în depozitul de date depinde foarte mult de procesul de curățare și procesul de transformare, și influențează totodată și rezultatele obținute în urma analizei.

1.4.2. Zona depozitului de date

În cadrul oricărei organizații sunt colectate și păstrate volume foarte mari de date cu scopul de a fi accesate, regăsite și analizate ulterior. În sistemele de tip BI&A principalele medii de stocare sunt depozitele de date.

Depozitul de date - reprezintă o bază de date, ce conține un volum mare de date, care este întreținută separat față de bazele de date operaționale ale companiei[21], oferind acces la toate datele organizației (tranzacții, produse, clienți, rezultate financiare, indicatori ai performanței și alte informații ce privesc afacerea). Principalul obiectiv al acestora este de a sprijini procesul

[12]Han, J., Kamber, M., Pei, J., Data mining: Concepts and Techniques, Editia a III-a, Morgan Kaufmann Publisher, San Francisco, 2012, p. 134.

decizional prin procesarea informațiilor, furnizând instrumente ce permit utilizatorilor finali accesarea rapidă a datelor conținute.

Principalele caracteristici ale unui depozit de date sunt[13]:

- *orientarea pe subiect* - este focalizat pe subiecte ale activității desfășurate de o organizație, subiecte ce necesită informații din mai multe surse pentru a reda o imagine completă a problemei cum ar fi: vânzări, clienți, profituri etc.
- *integrare* - este construit prin integrarea datelor provenite din mai multe surse: baze de date relaționale, fișiere etc.
- *caracter istoric* - datele stocate sunt date istorice. Acestea sunt ținute timp de 5-10 ani sau mai mult și sunt utilizate pentru analiza tendințelor, previziuni și comparații.
- *persistența datelor* - datele sunt stocate permanent și nu pot fi modificate, ceea ce înseamnă că actualizarea datelor din DW, ca urmare a modificării datelor sursă (externe și interne), se face prin adăugarea de noi date fără a modifica/șterge datele existente.
- Datele operaționale sunt agregate

Acesta conține diferite tipuri de date în funcție de cerințele informaționale ale utilizatorilor: date agregate sau sintetice, date detaliate etc., structurate astfel încât să permită utilizatorilor realizarea activităților de prelucrare și analiză, cum ar fi procesarea analitică online (OLAP), explorarea datelor, interogarea, raportarea etc.

În funcție de aria de cuprindere, depozitele de date pot fi clasificate în depozite de intreprindere (Enterprise Warehouse) și data mart[14,15]. De exemplu, un data mart (DM) este o versiune mai mică a depozitului de date, este specific unui grup de utilizatori, și conține doar o parte din volumul de date al întreprinderii în funcție de subiect sau departament, în timp ce depozitul de întreprindere stochează informații despre subiecte ce privesc întreaga intreprindere[16]. De obicei conține date agregate sau sintetizate dar și date detaliate.

---

[13] Han, J., Kamber, M., Data mining: Concepts and Techniques Editia II, Morgan Kaufmann Publisher, San Francisco, 2006, p. 106.

[14]Kilin, S., Review of modern business intelligence and analytics in 2015: How to tame the big data in practice/Case study – What kind of modern business intelligence and analytics strategy to choose?, disertatie, disponibila la https://aaltodoc.aalto.fi/bitstream/handle/123456789/18349/master_Kulin_Samu_2015.pdf?sequence=1&isAllowed=y, 2015.

[15]Han, J., Kamber, M., Data mining: Concepts and Techniques Editia II, Morgan Kaufmann Publisher, San Francisco, 2006, p. 132.

[16] Han, J., Kamber, M., Pei, J., Data mining: Concepts and Techniques, Editia a III-a, Morgan Kaufmann Publisher, San Francisco, 2012.

1.4.3. Zona de analiză sau zona de front-end

Dezvoltarea și viitorul organizației este influențat de deciziile adoptate de manageri de top, iar instrumentele utilizate de aceștia, pentru accesarea și utilizarea datelor stocate în DW, trebuie să fie performante.

Gestionarea datelor și informațiilor se face cu ajutorul tehnologiilor dedicate încorporate, care permit, pe de o parte, o mai bună înțelegere a trecutului, iar pe de altă parte, previzionarea viitorului, contribuind astfel la îmbunătățirea procesului decizional. Aceste componente dau valoare conceptului de BI&A și includ instrumente ce permite interogarea, analizarea și prezentarea informațiilor.

Zona de analiză sau componenta front-end include instrumente de accesare, extragere, interogare, analiză și prezentare a datelor ce pot fi împărțite în ***limbaje de procesare a datelor*** precum SQL și ***instrumente specializate*** cum ar fi instrumente OLAP (online analytical processing), instrumente de raportare, instrumente de explorare a datelor (data mining), ce permit transformarea datelor în informații și care dau valoare conceptului de BI&A.

Pentru a facilita procesul decizional, organizațiile au nevoie de instrumente eficiente care să le permită procesarea volumelor mari de date și transformarea acestora în informații utile. Studiile au demonstrat faptul că avantajele competitive pot fi obținute de organizații dacă adoptă decizii bazate pe date[17], iar sistemele de tip BI&A permit organizațiilor luarea unor decizii bazate pe date.

Instrumentele de analiză tipice incluse în cadrul sistemelor de tip BI&A sunt[18]:

Instrumentele de raportare - permit obținerea de rapoarte standard, preferate de 70% din utilizatori, și cuprind informații referitoare la performantele/rezultatele organizației, analiza retrospectiva a tendințelor, etc.

Analiza multidimensională - aplicații care utilizează o bază de date comună pentru a obține rapoarte statice în care utilizatorii au posibilitatea de a filtra raportul în funcție de criteriul dorit (aria geografică, produs etc.) sau rapoarte dinamice/navigabile (căutări, drill-down, consolidări, slicing, dicing) în care putem schimba nivelul de detaliere sau putem realiza căutări specifice.

Instrumentele de scorecards și dashboards - o formă a analizei multidimensionale comună în orice organizație, ce permite evaluarea rapidă a trendurilor, evenimentelor și

[17]Sivarajah, U., Kamal, M., Vishanth, Z.,Critical analysis of Big Data challenges and analytical methods, Journal of Business Research, Vol. 70, 2017.

[18]Williams, S., Business Intelligence Strategy and Big Data Analytics. A General Management Perspective, Morgan Kaufmann Cambridge, USA, 2016 p. 38.

rezultatelor cu rolul de a compara, monitoriza și îmbunătăți performanțele organizației. Acest tip de rapoarte, bazat pe excepții, permit compararea rezultatelor obținute cu rezultatele planificate.

Instrumente de interogare

Instrumentele de interogare permit utilizatorilor să obțină informații punctuale prin utilizarea limbajului SQL.

Datele asupra cărora utilizatori efectuează diferite analize sunt stocate în depozitul de date, care în majoritatea cazurilor este reprezentat de un sistem de gestiune a bazelor de date relațional și care permite atât colectarea și stocare, cât și interogarea. Limbajul SQL este un instrument de interogare puternic utilizat pentru extragerea și procesarea volumelor mari de date.

Instrumente data mining

Conceptul de "data mining" este definit ca fiind procesul de analiza a bazelor de date imense, de obicei depozite de date, cu scopul de a descoperi noi informații. Este un domeniu complex ce implică cunoștințe atât din domeniul informaticii, cât și cunoștințe din domeniul statisticii, învățare automată[19]. Termenul data mining este considerat a fi sinonim cu termenul Knowledge Discovery in Databases (KDD)[20, 21] - descoperirea cunoștințelor în baze de date. Data miningul este definit ca fiind procesul de extragere și analiză a datelor din baze de date sau depozite de date, cu scopul de a găsi noi informații, necunoscute anterior care să fie folosite în procesul decizional[22]. SPPS este un exemplu de aplicație ce oferă și capabilități de explorare a datelor. O alternativă la aplicațiile proprietare este mediul R ale cărui funcționalități pot fi extinse cu pachete ce permit analize statistice complexe, vizualizarea datelor etc.

1.5. Avantajele și dezavantajele sistemelor de tip BI&A

Sistemele de procesare a datelor au evoluat de la sisteme autonome la sisteme complexe de tip BI&A ce prezintă capabilități de colectare și integrare a datelor din diferite surse, pregătirea datelor pentru procesul de analiză, asigurând totodată și instrumentele necesare procesării acestora, permițând analiza avantajelor și dezavantajelor diferitelor opțiuni, astfel încât

[19]Alazemi, A., R., Data, text, and web mining for business intelligence: a survey, International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.3, No.2, 2013.
[20]Raisinghani, M., Business Intelligence in the Digital Economy: Opportunities, Limitations and Risks, Idea Group Inc, University of Dallas, USA, 2004, p. 37.
[21]Turnban, E., Aronson, J., E., Liang, T., P., Decision support systems and intelligent systems, Editia a 7-a, Pearson Education, Inc., New Jersey, 2007, p. 274.
[22]Turnban, E., Aronson, J., E., Liang, T., P., Decision support systems and intelligent systems, Editia a 7-a, Pearson Education, Inc., New Jersey, 2007, p. 263.

deciziile să fie luate pe baza unor date de calitate, precise și oportune[23]. Prin urmare, dintre avantajele pe care le prezintă sistemele de tip BI&A amintim[24]:

- Asigură accesul utilizatorilor la toate resursele informaționale necesare în procesul decizional
- Pentru anumite domenii de activitate oferă o serie de soluții predefinite
- Fiind considerat un sistem umbrelă, acesta va folosi întotdeauna tehnologiile cele mai noi și mai performante
- Este focalizat pe accesarea și distribuirea informațiilor importante către factori decizionali
- Permite vizualizarea indicatorilor de performanță
- Permite vizualizarea datelor sub formă de grafice atractive ușor de înțeles etc.

Provocările întâmpinate de organizații în ceea ce privește sistemele de tip BI&A sunt legate, pe de o parte de lipsa personalului specializat în prelucrarea datelor, iar pe de altă parte de faptul că accesul la toate datele valoroase ale companiei este destul de dificil, respectiv acestea nu vor fi utilizate eficient în procesul decizional.

Furnizorii de sisteme de tip BI

Dezvoltarea sistemelor de tip BI&A se datorează necesității facilitării procesului de analiza a datelor colectate de o organizație. Sistemele de tip BI includ foarte multe instrumente de la instrumente de căutare și regăsirea a datelor la instrumente de extragere și transformare, instrumente pentru utilizarea algoritmilor data mining, instrumente pentru vizualizarea datelor si diseminare a rezultatelor.

Printre furnizori de sisteme de tip BI ce includ instrumente de analiza a datelor precum limbajul de programare R enumerăm: Alteryx, KNIME, Microsoft, Revolution Analytics, Salford Systems, SAS, Tibco Software, etc.

Acești furnizori de sisteme de tip BI aduc laolaltă cele mai bune componente (instrumente ETL, tehnologii de stocare a datelor, instrumente de pregătirea datelor, instrumente de analiza și prezentare etc.), fiind integrat, de asemenea, și mediul R. De asemenea, aceștia au anticipat creșterea popularității mediului R, dar și beneficiile și limitările acestuia, deoarece utilizarea acestuia presupune cunoașterea limbajului R. Tehnicile de analiză și vizualizare a datelor fiind disponibile prin intermediul mediului R.

---

[23]Nagy, I., M., Proiectarea și Implementarea Depozitelor de Date pentru Business Intelligence aplicate în Economie, Teza de doctorat - rezumat, Cluj-Napoca, 2012.

[24]Gaille, B., 14 Pros and Cons of Business Intelligence, 2016, https://brandongaille.com/14-pros-and-cons -of-business-intelligence/, accesat la 27.02.2019.

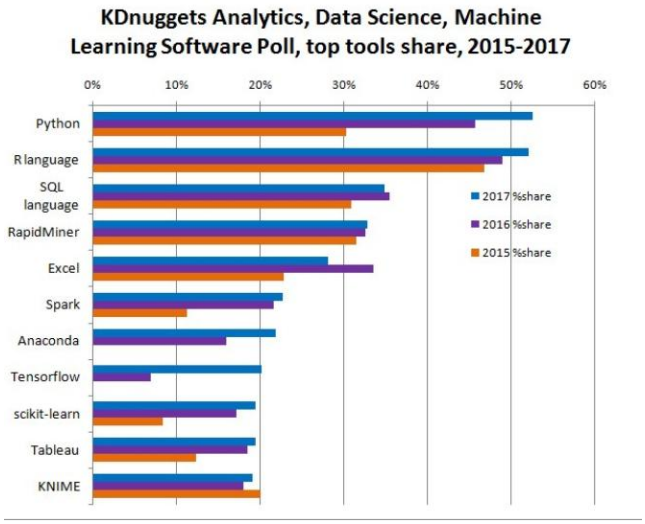


Figura 1. Cele mai utilizate instrumente de analiza a datelor din domeniul științei datelor în anul 2015 - 2017 [sursa: https://www.kdnuggets.com/].

Un studiu privind instrumentele utilizate în domeniul analizei datelor a fost realizat de cei de la KDNuggets (Figura 1) care arată că cele mai utilizate/populare instrumente de procesare a datelor sunt Python, limbajul R și limbajul SQL[25].

Studiul este susținut și de O'Reilly (realizat pe un eșantion de 900 de respondenți care au răspuns la un chestionar cu 64 cu întrebări legate de activități de analiză a datelor, instrumente utilizate, salar etc.) și care confirmă faptul că cele mai utilizate limbaje de programare sunt SQL (70%), R (57%) și Python (54%)[26].

[25]Piatetsky, G., New Leader, Trends, and Surprises in Analytics, Data Science, Machine Learning Software Poll, 2017, disponibil la https://www.kdnuggets.com/2017/05/poll-analytics-data-science-machine-learning -software-leaders.html, accesat la 12.03.2019.

[26]O'Reilly, Data Science Salary Survey results, 2016, disponibil la https://blog.revolutionanalytics.com/popularity/page/2/, accesat la 12.03.2019.

Capitolul II Limbaje de procesare a datelor utilizate în accesarea și prelucrarea datelor: Limbajul de procesare SQL și limbajul de procesare R

Prin valoarea pe care o conțin, datele colectate de companii sunt considerate a fi o adevărată mină de aur, iar procesul de transformare a acestora în informații joacă un rol important în cadrul organizațiilor conducând la îmbunatățirea performanței, obținerea de avantaje competitive și atingerea obiectivelor propuse. Procesarea acestor volume de date presupune, pe lângă instrumentele de analiză convenționale și utilizarea unor limbaje de procesare precum SQL, sau R în cazul în care dorim să facem o analiză mai avansată. Limbajele SQL respectiv R sunt utilizate în sistemele de tip BI&A pentru accesarea și procesarea volumelor mari de date.

În cadrul acestui capitol vom face o analiză mai detaliată a limbajului de procesare a datelor SQL respectiv a limbajului R utilizate în cadrul sistemelor de tip BI. Cu ajutorul acestora utilizatorii pot accesa și procesa volume mari de date într-o maniera interactiva și ad-hoc.

2.1. Limbajul de procesare a datelor R. Pachetul "Tidyverse"

Există foarte multe aplicații statistice dedicate analizei datelor. Dintre cele mai populare produse proprietare amintim: SPSS, SAS etc. Acestea au o interfață prietenoasă și oferă utilizatorilor finali o gamă largă de funcții statistice și opțiuni. Dezavantajul acestor instrumente de analiză este dat de costurile de achiziție a licenței, destul de ridicat, făcând dificilă adoptarea lor de către firmele mici și mijlocii.

În ultimii ani dezvoltarea limbajului R a fost extraordinară, acesta fiind folosit nu doar de mediul academic, ci și de mediul de afaceri. De asemenea, se observă că din ce în ce mai multe firme adoptă software-uri statistice open-sourse (în principal mediul R) pentru procesarea și analizarea datelor[27].

2.1.1. Caracteristicile R-ului

R este atât un limbaj de procesare cât și un mediu de dezvoltare ale cărui funcționalități pot fi extinse cu ajutorul pachetelor sau librăriilor, special dezvoltate pentru a rezolva anumite probleme, și care pot fi descărcate de pe site-ul oficial https://cloud.r-project.org/[28].

Cea mai recentă versiune a limbajului R este R 3.5.2[29, 30] și a fost lansată pe 20 decembrie 2018. Comparativ cu versiunile anterioare, îmbunatățirile aduse versiunii curente includ doar

[27] Fotache, M, Strimbei, C., SQL and data analysis. Some implications for data analysits and higher education, Procedia Economics and Finance 20, 2015

[28] https://cloud.r-project.org/, accesat la 19.02.2019.

rezolvarea unor probleme (bug fixes) legate de compatibilitatea unor scripturi și pachete cu noua versiune[31].

Conceput de statisticieni, pentru statisticieni, R permite, pe lângă manipularea datelor și realizarea de analize statistice, și construirea de grafice sofisticate, iar numele limbajului este dat de inițiala prenumelui celor care l-au creat, Ross Ihaka și Robert Gentleman[32].

R este alcătuit dintr-o serie de instrumente de analiză, vizualizare și modelare a datelor organizate sub formă de biblioteci sau librării de funcții numite *pachete*. *Pachetele* sunt colecții de funcții R și seturi de date. R conține două tipuri de pachete[33]: *pachete de bază* - care sunt automat încărcate în memoria locală atunci când RStudio este lansat în execuție (setul de pachete standard cu care vine R include: base, datasets, utils, graphics, states și methodes și furnizează o gamă largă de funcții și seturi de date disponibile implicit[34]); și *pachete contributive* - disponibile pentru descărcare și instalare, pe CRAN, și care trebuie încărcate în memorie pentru a putea fi utilizate.

Conectarea la baza de date

Accesul la baza de date dintr-un SGBD se realizează prin intermediul unei interfețe (DBI) între mediu R și SGBD. Conectarea la sursa de date în vederea accesării datelor sau informațiilor din baza de date "Închirieri mașini" se va face cu ajutorul pachetului "RPostgreSQL" se face cu ajutorul comenzii:

```
>install.packages('RPostgreSQL')
>library(RPostgreSQL)
>drv <- dbDriver("PostgreSQL")
library("DBI", lib.loc="~/R/win-library/3.4")
con <- dbConnect(drv,  dbname='BD Licenta', user='postgres', password='***')
                                                                    #conectare BD
dbListTables(con)                                    #listare tabele sub forma unui vector
>nume_obiectBD <- dbReadTable(con, "nume_obiectBD")
```

Interfața grafică a mediului R

Mediile de dezvoltare integrate dedicate limbajului R ușurează scrierea scripturilor, fiind intuitive și ușor de utilizat, și având o interfața mai prietenoasă.

---

[29] https://cran.r-project.org/bin/windows/base/, accesat la 22.02.2019.
[30] https://stat.ethz.ch/pipermail/r-announce/2018/000634.html, accesat la 22.02.2019.
[31] https://cloud.r-project.org/, accesat la 22.02.2019.
[32] Peng, R., D., R Programming for Data Science, 07-20-2015, https://www.cs.upc.edu/~robert/teaching/estadistica/rprogramming.pdf, accesat la 23.02.2019.
[33] Dusa, A., Oancea, B., Caragea, N., Alexandru, C., Jula, N., M., Dobre, A., M., R cu aplicații în statistică, Ed. Universității din București, 2015, p. 12.
[34] Kabacoff, R., I., R in Action. Data analysis and graphics with R, Manning Publications Co., New York, 2011, p. 15.

Un astfel de exemplu este RStudio, disponibil pentru toate sistemele de operare, și care este foarte util în manipularea și analiza datelor. Fereastra afișată utilizatorului este structurată în patru zone: editorul de comenzi, consola, spațiul de lucru, istoricul și fișierele, spațiu de vizualizare al pachetelor, al documentației și al graficelor.

Există foarte multe instrumente de analiză a datelor disponibile, iar alegerea unui anumit instrument de analiză a datelor este influențată de o serie de factori, precum: facilitățile oferite de aplicația respectivă, prețul, ușurința de învățare și utilizare, modul de prezentare a rezultatelor și lista poate continua.

R este unul dintre cele mai populare instrumente de analiză a datelor dezvoltat de statisticieni și care este în mod continuu îmbunătățit și susținut de o comunitate mare de cercetători și nu numai din mediu academic și mediu de afaceri.

Instalarea și încărcarea în memorie a pachetului "Tidyverse"

Instalarea pachetului Tidyverse se realizează prin comanda:

```
>install.packages("tidyverse")
```

Instalarea pachetelor se face o singură dată cu ajutorul funcției install.packages(), însă pentru a beneficia de îmbunatățirile aduse de autor folosim comanda update.packages() pentru actualizarea pachetelor deja instalate. În plus, pentru a putea fi utilizat în cadrul sesiunii de lucru acesta trebuie încărcat în memorie cu ajutorul funcției library() sau require():

```
> library(tidyverse)
>require(tidyverse)
>library()
>search()
```

Funcția search() permite afișarea pachetelor încărcate în memoria sesiunii de lucru. Eliminarea din memorie a unui pachet se realizează cu ajutorul funcție detach("package:tidyverse").

Pachetele conțin, pe lângă funcții, elemente și informații legate de autor, versiune, licența, descriere, documentație (prin comanda: help(tidyverse)) și seturi de date (prin comanda: data(package = "tidyverse")). Pentru a vedea detalii legate de pachetele instalate folosim comanda installed.packages() care listează pachetele instalate, versiunea lor, dacă este dependent sau nu de alte pachete, și alte informații[35].

[35] Kabacoff, R., I., R in Action. Data analysis and graphics with R, Manning Publications Co., New York, 2011, p. 16.

2.1.2. Descrierea pachetului Tidyverse

Pachetul "Tidyverse" este o colecție de pachete ce permit importarea, manipularea, explorarea, vizualizarea și modelarea datelor într-o manieră mai facilă[36]. Acesta este format din urmatoarele pachete fundamentale[37]: ggplot2, tibble, tidyr, reade, purrr, dplyr, forcats și string. ***Pachetul "ggplot2"*** este unul dintre cele mai elegante și versatile pachete de generare a graficelor. Implementează gramatica graficelor (grammar of graphics), o modalitatea coerentă pentru descrierea și construirea graficelor[38].

Reprezentarea grafică sau vizualizarea este un instrument important pentru obținerea de informații, însă datele nu sunt întotdeauna în forma dorită motiv pentru care sunt necesare și alte prelucrări. Pentru aducerea datelor în forma dorită sunt utilizate și alte pachete special proiectate pentru procesarea datelor.

Pachetul "tidyr" oferă un set de funcții care permit aducerea datelor în forma "tidy" (Figura 2). Datele în formă "tidy" sunt date aduse într-o formă consistentă, pe scurt fiecare variabilă se găsește într-o coloană/fiecare coloană este o variabilă, fiecare rând este o observabilă și fiecare set de date reprezinta un tabel[39, 40].

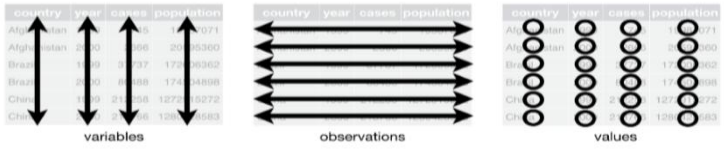


Figura 2. Caracteristicile datelor în formă "tidy" [sursa41:J. Porzak, 2017].

Pachetul "readr" furnizează modalități de citire a datelor din tabele (ex. csv etc.) într-o maniera rapidă și ușoară. Este special proiectat pentru a transforma diferite tipuri de date[42].
Pachetul "purrr" îmbunătățește funcționalitatea R-ului furnizând un set de instrumente pentru lucru cu vectori și funcții[43]. ***Pachetul "stringr"*** furnizează un set de funcții special proiectate pentru lucrul cu șiruri de caractere sau stringuri[44]. ***Pachetul "forcats"*** oferă o suită de funcții ce

[36]Bashir, S., Getting started in R – Tidyverse Edition, 2019, http://ilustat.com/shared/_Getting-Started-in-R.pdf, accesat la 28.03.2019.
[37]***Tidyverse packages, https://www.tidyverse.org/packages/, accesat la 17.03.2019.
[38]***Overview, https://ggplot2.tidyverse.org/, accesat la 18.03.2019.
[39]***The "Tidyverse", http://www.araastat.com/BIOF339_PracticalR/Lectures/lecture_tidyverse.pdf, accesat la 28.03.2019.
[40]***https://tidyr.tidyverse.org/, accesat la 18.03.2019.
[41] Porzak, J., Data Wrangling in the Tidyverse 21st Century R, prezentare, 2017, https://ds4ci.files.wordpress.com/2017/04/ds-portugal-april-2017.pdf, accesat la 01.04.2019.
[42]***https://readr.tidyverse.org/, accesat la 18.03.2019.
[43]***https://purrr.tidyverse.org/, accesat la 18.03.2019.
[44]***https://stringr.tidyverse.org/, accesat la 18.03.2019.

rezolvă cele mai frecvente probleme ale factorilor. Pentru a manevra variabilele categoriale (variabile care au un set posibil de valori cunoscute și fixe) R utilizează factors[45].

Tibbles sunt data.frame-uri, însă sunt ușor modificate pentru a funcționa mai bine cu pachetul "Tidyverse". Seturile de date/tibble prezintă următoarele caracteristici: nu pot fi modificate numele variabilelor/coloanelor, nu se poate converti șirurile de caractere în factori, nu permite potrivirea parțială a numelor, rândurile nu pot fi denumite[46]. Cele mai utilizate funcții ale pachetului "Tibbles" sunt[47,48]: glimpse() – care oferă o scurtă descriere a setului de date fiind similară cu funcția str(), as_tibble() ce permite convertirea unui data.frame în tibble, pentru crearea de seturi de date de tip tibble folosim funcția tibble() sau tribble(), pentru adăugarea de rânduri și coloane în cadrul unui set de date folosim funcțiile add_row() și add_column() etc.

Pachetul "dplyr" furnizează funcții ce permit manipularea datelor sau un set de "verbe" cu ajutorul cărora putem rezolva cele mai uzuale cerințe legate de manipularea și transformarea datelor permițând crearea de noi variabile, sumarizarea variabilelor, redenumirea lor, ordonarea observațiilor etc. Principalele funcții ale ***pachetului "dplyr"*** ce permit procesarea datelor sunt: *filter()* permite filtrarea observabilelor/rândurilor, *arrange()* - permite ordonarea observabilelor, *select()* - permite selectarea variabilelor, *mutate()* - permite realizarea de noi variabile, *summarise()* - reduce un set de date la o singură valoare[49] - un fel de funcție agregat, *group_by()* - care modifică scopul fiecărei funcții de la operarea asupra întregului set de date la operarea asupra unor grupuri de rânduri. Aceste funcții sunt numite verbe ale limbajului de manipulare a datelor. Toate funcțiile funcționează similar, iau ca argument un data.frame, al II-lea argument descrie ce se face cu data.frame-ul prin utilizarea numelui variabilelor, iar rezultatul este un nou data.frame. Dacă dorim să salvăm rezultatul va trebui să folosim operatorul de atribuire <-. Pipe (%>%) permite combinarea mai multor operații, iar modul de scriere a interogărilor este focalizat pe transformare și nu pe ceea ce este transformat.

---

[45]***https://forcats.tidyverse.org/, accesat la 18.03.2019.

[46]Erhardt, E., B., Introduction into R tidyverse, prezentare, 2018, https://statacumen.com/teach/ShortCourses/R_Programming/IntroTidyverse_ACASA_201802/Erhardt_RTidyverse_20180216.pdf, accesat la 28.03.2019.

[47]Wickham, H., Package 'tibble', 2019, https://cran.r-project.org/web/packages/tibble/tibble.pdf, accesat la 28.03.2019.

[48]Wickham, H., Grolemund, G., R for data science. Import, Tidy, transform, visualize and model data, Published by O'Reilly Media, Inc., Canada, 2016, p. 75.

[49]***https://dplyr.tidyverse.org/, accesat la 18.03.2019.

2.1.3. Avantajele mediului R/Caracteristicile de bază ale R-ului

O caracteristică deloc de neglijat este faptul că R este open-source, distribuit sub licență GNU - General Public Licence, în mod gratuit, ceea ce înseamna că orice utilizator poate prelua codul sursă al programelor, îl poate studia, modifica, îmbunătăți, utiliza, copia, distribui sau se poate folosi parți din codul sursă în alte programe ce pot fi ulterior comercializate.

Software-urile open-source, conform filosofiei GNU, sunt caracterizate de libertatea oferită utilizatorilor săi de a-l studia, copia, utiliza, distribui, modifica și îmbunătăți prin accesul la codul sursă al programului[50]. Concret vorbim de patru forme de libertate oferite utilizatorilor[51]:

- Libertatea 0 - utilizatorii pot folosi programul în orice scop;
- Libertatea 1 - deoarece avem acces la codul sursă oricine poate studia modul de funcționare și poate adapta programul în funcție de nevoi;
- Libertatea 2 - oricine poate distribui copii, în scopul ajutării aproapelui sau;
- Libertatea 3 - accesul la codul sursa ne permite sa aducem o serie de îmbunatățiri programului, iar versiunea îmbunătățita poate fi pusă la dispoziția publicului.

Fiind un software open-source, acesta prezintă și o serie de avantaje competitive față de alte aplicații proprietare, cum ar fi: SPSS, SAS etc, întrucât dispar costurile alocate licenței. Datorită acestui lucru, un număr din ce în ce mai mare de companii încep sa folosească, ca instrument de analiză a datelor, limbajul R.

Flexibilitatea limbajului și reproductibilitatea analizelor sunt alte caracteristici pe care limbajul R le posedă.

Funcționalitatea R-ului pot fi extinsă prin intermediul pachetelor cu ajutorul cărora pot fi realizate diverse analize a datelor, de la analize statistice la previziuni și reprezentări grafice. În prezent, R are foarte multi utilizatori, peste 2 milioane la nivel mondial, și mii de utilizatori ce contribuie la dezvoltarea lui[52] oferind totodată și suport tehnic.

R poate rula pe majoritatea platformelor importante și este disponibil pentru majoritatea sistemelor de operare. Fiind open-source oricine este liber să adapteze, modifice sau să îmbunătățească codul sursă sau să adapteze software-ul astfel încât acesta să devină compatibil și

[50]***Open source software, https://www.onebusiness.ca/sites/default/files/MEDI_Booklet_Open_Source_Software_accessible_E.pdf, accesat la 22.02.2019.

[51]***https://www.gnu.org/philosophy/philosophy.en.html, accesat la 22.02.2019.

[52] Caragea, N., Alexandru, C., A., Dobre, A., M., Bringing new opportunities to develop statistical software and data analysis tools in Romania, MPRA nr. 48772, 2013.

cu alte platforme dorite. Prin urmare, R poate rula pe tableta, telefoane și console pentru jocuri. O altă caracteristică a R-ului este dată de capabilitățile grafice sofisticate.

Un alt avantaj este dat de faptul că există foarte multe cărți și resurse disponibile ce permit învățarea limbajului R.

2.1.4. Limitele limbajului de procesare a datelor R

La ora actuala, nu exista limbaj de procesare a datelor sau sistem de analiza statistica perfect, iar limbajul R nu face excepție. Principalele limitări ale mediului R în ceea ce privește procesarea datelor sunt date de[53]:

- ***colecțiile de date sau datele***, stocate de obicei în baze de date relaționale (PostgreSQL etc.), fișiere etc. sunt accesate din R. Instrumentele statistice de obicei încarcă/importă datele ce urmează a fi prelucrate fie prin importarea directa a datelor din fișiere externe prin intermediul meniurilor, fie prin intermediul pachetelor dedicate ce permit interogarea bazei de date în mod direct și importă rezultatele obținute în spațiul de lucru R pentru procesări ulterioare.
- ***interfața*** - nu este foarte prietenoasă cu utilizatorii. Chiar dacă există și software-uri open-source ce îmbunătățesc dialogul cu mediul R (ex.: RStudio), interfața pe care mediul R o are se bazează pe linii de comandă și pe scripturi fiind puternic orientat spre programare.
- ***sintaxa limbajului*** - sintaxa limbajului R este relativ dificil de învățat comparativ cu alte software similare etc.

În ceea ce privește procesarea datelor, cea mai importantă limitare a mediului R este dată de modul de lucru cu datele, deoarece acesta încarcă toate datele ce urmează a fi procesate în memoria RAM, ceea ce înseamnă volumul de date prelucrate este limitat la spațiul de memorie RAM disponibil. O soluție ar fi utilizarea unor calculatoare ce dispun de memorie RAM mai mare sau utilizarea unor pachete dedicate ce permit procesarea volumelor mari de date (precum: ff, bigmemory etc.) ce depășesc capacitatea memoriei RAM[54].

2.2. Limbajul de procesare a datelor SQL

Bazele de date, o tehnologie inventată în ani ‘70, nu mai sunt doar simple platforme de stocare a datelor, ci încet încet încep să devină motoare de transformare a datelor în informații

[53] Fotache, M, Strimbei, C., SQL and data analysis. Some implications for data analysits and higher education, Procedia Economics and Finance 20, 2015.

[54] Dusa, A., Oancea, B., Caragea, N., Alexandru, C., Jula, N., M., Dobre, A., M., R cu aplicații in statistica, Ed. Universității din București, 2015, p. 195.

despre clienți, produse și practici comerciale etc. fiind instrumente potrivite pentru prelucrarea și extragerea informațiilor importante. Scopul oricărei intreprinderi este de a găsi informații importante ascunse în spatele volumelor de date colectate și care sunt de ordinul gigabytes sau terrabytes. Limbajul SQL permite extragerea informațiilor din bazele de date relaționale.

2.2.1. Caracteristicile limbajului SQL

Bazele de date au fost concepute pentru a stoca și prelucra volume mari de date, iar limbajul SQL este limbajul standardizat de interogare ce permite interacțiunea cu o baza de date. Limbajul SQL (acronim ce vine de la Structered Query Language) este un limbaj dedicat bazelor de date și cuprinde comenzi atât pentru declararea structurii bazei de date cât și pentru consultarea și editarea conținutului, respectiv pentru crearea și administrarea conturilor de utilizatori/grup de utilizatori, fiind mai mult decât un simplu instrument de consultare a bazelor de date. Astfel, limbajele de programare dedicate bazelor de date, SQL, sunt formate din[55]: limbaje de manipulare a datelor (DML - Data Manipulation Language) - care permit realizarea de operații CRUD (de consultare/citire, scriere, modificare și ștergere a datelor), limbaje de definire a datelor (DDL - Data Definition Language) - permite definirea schemei bazei de date și limbaje de control al datelor (DCL - Data Control Language) - care permit administratorilor de baze de date să acorde drepturi de acces la baza de date, iar după uni autori[56] (D.D. Chamberlin si R.F. Boyce) ar fi si limbajul de interogare al datelor (DQL - Data Query Langauge).

Adoptarea pe scară largă a limbajul SQL se datorează faptului că[57] acesta este un limbaj neprocedural deoarece utilizatorul definește rezultatul dorit, iar SGBD-ul se ocupă de căutarea și extragerea datelor; este un limbaj închis; este relativ simplu de învățat și înțeles (comenzile se scriu în limba engleză); este un limbaj standardizat dedicat BD elaborat de ANSI (American National Standards Institute) și ISO (International Organization for Standardization - numărul alocat este 9075).

Deși, inițial limbajul SQL a fost gândit pentru bazele de date relaționale tradiționale, practica a demonstrat că acesta este un limbaj eficient ce poate fi implementat cu succes și în noile tipuri de sisteme de baze de date, în particular în sistemele de gestiune de tip Big Data (Big

[55]Anuradha, G., Joel Varma, D., Graph Mining Extensions in Postgresql, Indian Journal of Science and Technology, Vol 9(35), 2016.
[56]***https://data-flair.training/blogs/sql-tutorials-home/#interview-questions, accesat la 20.02.2019.
[57]Silva, Y., N., Almeida, I., Queiroz, M., SQL: From traditional database to Big Data, SIGCSE'16, Memphis, Tennessee, USA, 2016.

Data Management Systems - BDMS), ce procesează rapid volume imense de date provenite din surse variate.

Apariția sistemelor de gestiune de tip Big Data au schimbat structura pieței SGBD-urilor, sistemele de gestiune a bazelor de date relaționale tradiționale ocupând în prezent doar o fracțiune din aceasta. Popularitatea limbajului SQL, supranumit de M. Stonebraker și "limbajul intergalactic"[58], se datorează în principal faptului că este un limbaj de nivel înalt cu o sintaxa ușor de înțeles fiind implementat pe majoritatea sistemelor de gestiune a bazelor de date, de la cele proprietare la cele open-source[59], iar adoptarea sa pe scară largă facilitând standardizarea acestuia. Cei mai cunoscuți producători de SGBD-uri proprietare din domeniul afacerilor sunt: Oracle, IBM's DB2 și Microsoft SQL Server. Dintre SGBD-urile open-source cele mai cunoscute sunt: PostgreSQL și MySQL. Toate aceste SGBD-uri au implementat limbajul SQL, sau mai corect un dialect al limbajului SQL (existând o serie de diferențe în ceea ce privește sintaxa) ca limbaj de interogare.

Atenția va fi focalizată pe utilizarea limbajului SQL, fiind un limbaj orientat pe tupluri, pentru procesarea și analizarea datelor, ceea ce presupune utilizarea limbajelor de manipulare a datelor pentru a extrage diferite informații din datele colectate (considerat a fi un proces dificil ce necesita timp) de o organizație.

Gestiunea volumelor mari de date presupune realizarea a trei tipuri de operațiuni: colectarea datelor, stocarea datelor și prelucrarea datelor. Prin urmare, componentele principale pentru analiza volumelor mari de date sunt bazele de date și instrumentele de analiză statistică.

Procesarea acestor volume de date presupun utilizarea limbajelor de procesare a datelor SQL pentru analizarea acestora: fraza/comanda SELECT care include clauze și funcții pentru găsirea valorilor minime (MIN()), maxime (MAX()), agregări pentru realizarea de totaluri (SUM()), pentru a număra înregistrările ce îndeplinesc o anumită condiție (COUNT()), valorile medii (AVG()), grupări (GROUP BY), înregistrările care îndeplinesc o anumită condiție (HAVING) care nu presupune altceva decât interogarea bazelor de date[60].

[58] Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009, p. 73.
[59] Fotache, M, Data Processing Languages for Business Intelligence. SQL vs. R, Informatica Economică vol. 14, nr. 1, 2016.
[60] Fotache, M, Strimbei, C., SQL and data analysis. Some implications for data analysits and higher education, Procedia Economics and Finance 20, 2015.

2.2.2. Descrierea bazei de date analizate

Tot mai multe organizații acordă o importanță sporită informatizării, iar bazele de date ale acestora conțin informații importante legate de produsele sau serviciile oferite, furnizori, activitatea financiar-contabilă, clienți etc.

Bazele de date reprezintă una dintre componentele principale ale unui sistem informatic, ce au ca scop stocarea și prelucrarea unui volum mare de informații în funcție de nevoile utilizatorilor, putând fi definită ca fiind: *o colecție de date aflate în interdependență, împreună cu descrierea datelor și a relațiilor dintre ele*, sau *ca o colecție de date utilizată într-o organizație, persistentă, partajată, definită riguros și controlată la nivel central*[61].

Două aspecte trebuie avute în vedere atunci când vorbim de o baza de date, și anume: *schema sau structura* bazei de date care este partea statică, și arată modul de organizare a bazei de date; și *conținutul* care este partea dinamică, și reprezintă ansamblul informațiilor stocate la un moment dat în baza de date.

Principalele funcții ale unui sistem de gestiune a bazelor de date relaționale sunt[62]: descrierea datelor la nivel fizic și conceptual; crearea și exploatarea BD; controlul integrității; confidențialitatea informațiilor conținute; acces simultan al unui număr mare de utilizatori; securitatea în funcționare; monitorizarea performantelor; restructurarea și revizia etc.

SGBD-urile, proprietare (cum ar fi: Oracle, Microsoft SQL etc.) sau open-source (PostgreSQL, MySQL etc.) pot fi privite și ca instrumente ce permit interogarea și procesarea volumelor de date colectate. Datele pot fi extrase din diferite surse de date cu ajutorul limbajului SQL (folosind interogări SQL). Pentru realizarea unor analize mai complexe (analize descriptive, analize predictive, reprezentări grafice etc.) este necesară utilizarea unor instrumente de analiză specializate, precum SPSS, R, SAS etc.

PostgreSQL este un server de baze de date sau sistem de gestiune a bazelor de date relaționale - adică un software specializat pentru crearea, utilizarea, administrarea bazelor de date. Este important să precizăm faptul că PostgreSQL este un SGBD open-source, fiind clasat în topul SGBD-urilor open-source printre primele[63].

În contextul utilizării SGBD-urilor în dezvoltarea aplicațiilor, tranzacțiile joacă un rol important. Modul în care tranzacțiile sunt gestionate de SGBD poate influența performanța și

[61]Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009, pp. 20.
[62]Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009, pp. 19.
[63]Walker, A., 18 Best Open-Source and Free Database Software, 2017, disponibil la https://learn.g2crowd.com/free-database-software, accesat la 19.02.2019.

complexitatea codurilor de programare ce utilizează serviciile SGBD-ului, fiind importantă cunoașterea principiilor din spatele tranzacțiilor din sistemele SQL.

Majoritatea SGBD-urilor respectă proprietățile ACID, acronim ce vine de la[64, 65]: *atomicitate, consistență, izolare* și *durabilitate*. ***Atomicitatea*** - se referă la faptul că fiecare comandă din cadrul unei tranzacții funcționează pe principiul totul sau nimic. În cazul în care nu obținem nici un mesaj de eroare, tranzacția se încheie prin comanda COMMIT ce consemnează modificările în BD. Comanda ROLLBACK anulează celelalte comenzi din tranzacție și este utilizată atunci când una din comenzile tranzacției eșuează. ***Consistența*** - se referă la faptul că tranzacțiile trebuie sa respecte restricțiile definite. ***Izolarea*** - se referă la faptul că la interogarea BD ceea ce obținem este ceea ce a fost în BD inanitate de inițierea oricărei tranzacții sau după terminarea oricărei tranzacții aflată în derulare. ***Durabilitatea*** - se referă la faptul că odată ce o tranzacție s-a încheiat cu succes (COMMIT), modificările sunt consemnate definitiv în BD.

În ultimul timp se constată că un număr din ce în ce mai mare de organizații adoptă sisteme open-source. Acest lucru este confirmat și de rezultatele raportului intitulat "State of the Open-Source DBMS Market, 2018" realizat de cei de la Gartner care arată că până în anul 2022, un procent mai mare de 70% din noile aplicații in-house vor fi dezvoltate pe un sistem de gestiune a bazelor de date relațional open-source (SGBDOS), și 50% din sistemele de gestiune a bazelor de date relaționale (SGBDR) comerciale existente vor fi înlocuite sau se vor afla într-un proces de înlocuire cu una din soluțiile open-source disponibile pe piață[66], costurile companiilor reducându-se astfel cu ~ 80%[67].

Punctele de închiriere a mașinilor și/sau alte produse ce nu pot fi transportate pe avion constituie un tip de activitate importantă mai ales în orașele dezvoltate unde avem aeroporturi, precum Iași, Cluj sau București etc. Baza de date "Închirieri mașini" realizată înregistrează informații ce trebuie stocate și gestionate prin intermediul acesteia despre: birourile de închiriere, angajați acestora, mașinile și/sau produsele disponibile pentru a fi închiriate, închirieri, restituiri, contracte, facturi, date despre clienți (Anexa 1 - Figura 1).

---

[64]Maksimov, D., Performance Comparison of MongoDB and PostgreSQL with JSON types, Tallinn university of technology, Faculty of Information Technology Department of Informatics, Disertatie, 2015, p. 11 - 12.

[65]Champagne, J., The Top 7 Free and Open Source Database Software Solutions, 2017, disponibil la https://blog.capterra.com/ free-database-software/, accesat la 19.02.2019.

[66]***https://mariadb.com/newsroom/press-releases/report-state-of-the-open-source-dbms-market-2018-by-gartner-includes -pricing-comparison-with-mariadb-2/, accesat la 19.02.2019.

[67]***https://www.splendiddata.com/2017/06/06/gartner-says-2018-70-new-applications-will-run-open-source-databases/, accesat la 19.02.2019.

În Anexa 1 - Figura 1 este prezentată structura bazei de date Închirieri mașini compusă din următoarele tabele: JUDETE, CODURI_POSTALE, BIROUINCHIRIERI, CLIENTI, DEPARTAMENTE, MANAGERI, ANGAJATI, CLASAAUTO, CONTRACT, RETURNAREAUTOTURISM, STATUSAUTOTURISM, PRODUSE_INCHIRIATE, FACTURI, DETALII_FACTURA, ASIGURARE, MARCAAUTOTURISM, MODELEAUTOTURISM.

Aceste tabele stochează, în principal, informații despre birourile de închiriere ale organizației, angajați și clienții acestora, modelele de mașini și/sau produsele puse la dispoziția clienților pentru a fi închiriate, contractele încheiate cu clienți, facturile emise clienților, dar și dacă mașinile închiriate au sau nu asigurare.

Funcții SQL utilizate pentru analiza datelor

Limbajul SQL permite extragerea din baza de date a unor seturi de înregistrări. Orice interogare/comandă SQL trebuie să conțină obligatoriu clauzele SELECT și FROM, însă poate include și elemente precum: funcții, clauze, expresii, predicate, subinterogări etc.

Rezultatele interogărilor/comenzilor SQL pot fi, de asemenea, salvate fie în baza de date (sub formă de tabele anonime temporare, tabele, tabele virtuale[68] etc.), fie pot fi exportate din SGBD în alte baze de date, fișiere excel, CSV, fișiere text etc.

Comanda/fraza SELECT nu permite doar extragerea și filtrarea înregistrările din baza de date, ci și realizarea a diferite procesări care se aplică la nivel de linie sau de grup cu ajutorul clauzelor GROUP BY și HAVING, sau funcțiilor statistice numite și funcții analitice[69] sau funcții de agregare statistică precum: SUM, COUNT, AVG, MIN, MAX. Obiectivul standardului SQL este analiza datelor prin intermediul funcțiilor OLAP. În ceea ce privește analiza datelor, PostgreSQL implementează pe lângă funcții statistice și funcții, precum[70]: RANK, CUME_DIST etc.

Majoritatea SGBD-urile open-source oferă funcții statistice extrem de utile în procesarea datelor precum[71]: STDEV_POP (deviația standard pentru o populație), STDEV_SAMP (abaterea standard pentru un eșantion), CORR[72] (corelație) etc.

---

[68] Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009, p. 176.
[69] Stonebraker, M., What Does 'big Data' Mean?, 2012, disponibil la https://cacm.acm.org/blogs/blog-cacm/155468 - what-does-big-data-mean/fulltext, accesat la 22.03.2019.
[70] ***https://www.postgresql.org/docs/9.5/functions-aggregate.html, accesat la 27.03.2019.
[71] ***https://www.postgresql.org/docs/9.5/functions-aggregate.html, accesat la 27.03.2019.
[72] ***The Pearson Correlation Coefficient Formula in SQL, https://chartio.com/learn/postgresql/correlation - coefficient-pearson/, accesat la 27.03.2019.

2.2.3. Avantajele și dezavantajele limbajului SQL

Accesarea, citirea și analiza datelor cu ajutorul instrumentelor statistice pentru analiză și exportarea rezultatelor în cadrul altor sisteme este o sarcină destul de dificilă ce necesită timp comparativ cu timpul alocat analizei în sine. În general, aplicațiile statistice, precum R-ul, nu sunt potrivite pentru manipularea volumelor mari de date. Spre deosebire de R, SGBD-urile, ca instrumente de procesare a datelor, sunt mai potrivite pentru manipularea și extragerea datelor din volume mari de date. Instrumentele analitice au o serie de funcții puternice însă sintaxa acestora este mai dificil de înțeles și învățat.

Limbajul SQL ce este implementat pe majoritatea SGBD-urilor, permite gestionarea și interogarea datelor din bazele de date relaționale. Cu toate acestea, majoritatea SGBD-urilor au implementat propriul dialect ce poate prezenta anumite limitări sau extensii. Extensiile includ de obicei funcții suplimentare ce permit utilizatorilor efectuarea unor operații mai complexe față de standardul SQL. Principalele avantaje ale sistemului de gestiune a bazelor de date PostgreSQL sunt[73]: spre deosebire de alte SGBD-uri, PostgreSQL se apropie cel mai mult de standardul SQL având implementate 160 de caracteristici din 179 ale standardului SQL; fiind un proiect open-source, codul sursă este dezvoltat de o comunitate de programatori foarte mare ce participă și la realizarea documentației; este compatibil cu principiile ACID asigurând integritatea datelor; este compatibil cu o gamă largă de limbaje de programare și platforme, ceea ce face ca migrarea bazelor de date de pe un sistem de operare pe altul sau integrarea cu alte instrumente să fie mai ușoară etc.

Popularitatea limbajului SQL se datorează faptului că permite accesarea și manipularea datelor stocate în SGBD, permite descrierea datelor, permite crearea/ștergerea/modificarea tabelelor, efectuarea operațiilor CRUD etc., în timp ce limbajul R este dedicat analizei și procesării datelor.

Deși la prima vedere cele două limbaje par a fi complet diferite și utilizate în scopuri diferite acestea au câteva caracteristici comune. Limbajul SQL este utilizat pentru interogarea și actualizarea datelor stocate în bazele de date relaționale, în timp ce limbajul R este utilizat pentru accesarea și analiza datelor.

Inițial limbajul SQL nu a fost proiectat pentru analiza datelor, ci mai degrabă pentru optimizarea, interogarea și actualizarea datelor. Analiza datelor cu ajutorul limbajului SQL

[73] Drake, M., SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems, 2019, disponibil la https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of- relational-database-management-systems, accesat la 03.04.2019.

necesită construirea de interogări complicate. Numărul joncțiunilor între tabele influențează timpul de răspuns/vitezei de răspuns a interogărilor lansate. Spre deosebire de SQL, limbajul R este dedicat analizei și procesării datelor.

Tabelul 1. Principalele diferențe și similitudini dintre SQL și R

Caracteristica	SQL	R
Originea	Este un limbaj de nivel înalt creat de IBM	Derivă din limbajul S și a fost creat de R. Ihaka și R. Gentleman
Scopul	Este un limbaj de nivel înalt creat pentru interogarea și actualizarea bazelor de date relaționale	Este un limbaj open-source dedicat procesării, analizei si vizualizării
Implementare	pe majoritatea SGBD-urilor	Poate fi descărcat și instalat de pe pagina oficială CRAN
Dialecte	Furnizori de SGBD-uri au implementat un standard SQL ușor diferit	Mai compact; există câteva dialecte comerciale (ex. Microsoft R)
Modularizare	Compact (nu avem librării independente). Implementarea este disponibilă prin intermediul noilor versiuni SGBD	Funcționalitățile R-ului pot fi extinse prin intermediul pachetelor disponibile pe CRAN, github etc.
Stocarea datelor	Utilizat doar de SGBDR, însă este adoptat și de alte tipuri de baze de date	Nu este disponibil
Volumul datelor procesate	Teoretic nelimitat, în practică depinde de baza de date, depozitul de date etc.	Limitat la memoria RAM disponibilă, însă există pachete dedicate pentru procesarea volumelor mari de date (ex. ff, bigmemory etc.)
Surse de date	Depinde de SGBD, fiind limitat la crearea BD cu ajutorul SGBD-ului	Putem accesa date de pe orice sursa de date
Procesarea datelor	Cel mai cunoscut limbaj de procesare a datelor	R vine cu o serie de pachete de bază ce primit prelucrarea datelor, dar fincționalitățile lui pot fi extinse cu alte pachete.
Funcțiile OLAP	Implementate în standardul SQL:2009	Implementate în pachetul dplyr
Tabele pivot	Prin intermediul interogărilor recursive sau folosind clauza *pivot* sau funcția crosstab	Implementate în pachetul tidyr
Vizualizarea datelor	--	Implementate în pachetul ggplot2
Analiza datelor	Include indicatori din statistica descriptivă dar și alte teste statistice	Nelimitat
Dificultatea de învățare	Ușor de învățat	Mai dificil de învățat

Preluat după[74]: M. Fotache, 2016.

[74]Fotache, M., Data Processing Languages for Business Intelligence. SQL vs. R, Informatica Economică vol. 20, nr. 1, 2016.

R-ul nu poate fi considerat o alternativă a sistemelor de gestiune a bazelor de date deoarece acesta încarcă toate datele ce urmează a fi procesate în memoria RAM, și nu stochează datele sau rezultatele prelucrării. De asemenea, R-ul nu permite optimizarea interogărilor ceea ce poate duce la scăderea vitezei de răspuns/creșterea timpului de răspuns.

Cele două limbaje de procesare, R și SQL, sunt utilizate în cazuri/situații diferite. Limbajul SQL este utilizat preponderent în stocarea datelor critice ale organizațiilor, permițând utilizatorilor accesarea, modificare, inserarea și ștergerea datelor într-un mediu centralizat, în timp ce limbajul R este utilizat pentru lucrul cu datele (procesarea, analizarea și vizualizarea)[75]. O parte din operații sunt mai ușor de realizat în SQL altele nu deoarece crește complexitatea interogărilor.

[75] Fotache, M., Data Processing Languages for Business Intelligence. SQL vs. R, Informatica Economică vol. 20, nr. 1, 2016.

Capitolul III Analiza comparativă a sintaxei interogărilor SQL și "Tidyverse"

Pentru procesarea și analizarea datelor, utilizarea limbajului SQL sau a limbajului R ar putea fi o soluție pentru identificarea și rezolvarea problemelor de afaceri. Însă problemele de afaceri trebuie transformate mai întâi în întrebări, iar răspunsurile la aceste întrebări pot fi obținute prin intermediul interogărilor realizate asupra bazei de date/depozit de date a organizației. De asemenea, prezentarea și interpretarea rezultatelor obținute în urma interogărilor reprezintă o etapă importantă în procesul de analiză. În cadrul capitolului se va face un studiu comparativ a modului în care cele două limbaje R și SQL procesează datele stocate în bazele de date/depozitele de date ale sistemelor de tip BI.

3.1. Funcții de procesare a datelor folosite în SQL și R

Compararea modului în care cele două limbaje R și SQL procesează datele stocate în depozitele de date ale sistemelor de tip BI, reprezintă principalul obiectiv al acestei lucrări. Prin urmare, în continuare vom face o comparație între principalele funcții din R și SQL utilizate frecvent în diferite tipuri de interogări din cadrul sistemelor de tip BI&A.

Toate interogările prezentate în cadrul lucrării au fost realizate pe baza de date "Închirieri mașini" a cărei structură este prezentată în Figura 1 (Anexa 1). Schema bazei de date analizate a fost creată în PostgreSQL, fiind populată cu date fictive.

Pentru a putea fi procesate cu ajutorul limbajului R, tabele create în PostgreSQL au fost importate în mediul R, ca data.frame-uri, cu ajutorul pachetului RPostgreSQL (ce conține o serie de funcții de interogare precum: dbSendQuery(), dbGetQuery(), dbReadTable(), collect(), dplyr()) prin comanda[76, 77]:

```
>if(!"RPostgres" %in% installed.packages()) {
      install.packages("RPostgres")}
>library(RPostgreSQL)
>con <- dbConnect(PostgreSQL(), user= "user", password="passwd",
dbname="BD Licenta")                                    #conectarea R-ului la serverul BD
>dtab = dbGetQuery(con, "modeleautoturism")     #citirea tabelului din baza de date
```

[76] Conway, J., Eddelbuettel, D., Nishiyama, T., Prayaga, S., K.,Tiffin, N., RPostgreSQL: R Interface to the 'PostgreSQL' Database System, 2017, disponibil la https://cran.r-project.org/web/packages/RPostgreSQL/index.html, accesat la 22.03.2019.

[77] ***Performance comparison: odbc vs RPostgreSQL, https://rpubs.com/nwstephens/334324, accesat la 22.03.2019.

```
>dbDisconnect(con)                    #deconectarea de la baza de date
```

R este potrivit pentru selectarea datelor din baza de date și nu pentru crearea de noi tabele în baza de date sau actualizarea datelor, însă dispune de funcții ce permit realizarea tuturor operațiilor. R conține, de asemenea, și comenzi pentru gestionarea tranzacțiilor (dbBegin(), dbRollback(), dbCommit()), esențiale atunci când vrem să vedem dacă un bloc de comenzii SQL (operații CRUD) sunt executate corect înainte de a fi înregistrate permanent[78].

3.2. Interogări selecție, proiecție și joncțiune

Pentru realizarea interogărilor vom folosi pachetele incluse în "Tidyverse", "dplyr" și "tidyr", ce permit procesarea datelor în R. Vom încerca să potrivim sintaxa interogărilor SQL cu sintaxa interogărilor scrise în R pentru a putea face un studiu comparativ în ceea ce privește sintaxa celor două limbaje de procesare a datelor.

În R, interogarea este împărțită în mai multe etape, iar rezultatul fiecărei etape este un data.frame sau tabel care este preluat și procesat în continuare în celelalte etape. Deși la prima vedere sintaxa pare simplă, comparativ cu standardul SQL uneori soluțiile sunt mai dificil de găsit.

Interogarea "*Afișarea informațiilor privind nr. facturii, poziția pe factură și codul produselor regăsite pe facturile cu numărul cuprins intre 101 - 109?*" necesită utilizarea operațiunilor de bază precum selecție, proiecție și sortarea/ordonarea rezultatului obținut după numărul facturi. Prin urmare, sintaxa interogării scrisă în cele două limbaje de procesare a datelor, R respectiv SQL, este:

SQL:	**R:**
SELECT NRFACTURA, LINIEFACTURA, CODPROD FROM DETALII_FACTURA WHERE NRFACTURA BETWEEN 100 AND 109 ORDER BY NRFACTURA;	R96 <- detalii_factura%>% filter(nrfactura >= 100 & nrfactura <= 109)%>% select(nrfactura, liniefactura,codprod)%>% arrange(nrfactura) R96
Rezultatul interogării este prezentat în Anexa 2 Figura 1.	

În R manipularea datelor se face prin utilizarea verbelor sau funcțiilor select(), filter(), select() și arrange() oferite de pachetul "dplyr" din "Tidyverse" ce permit selectarea înregistrărilor din tabela detalii_factura ce îndeplinesc condiția din funcția filter(). Ordonarea

[78] Mainmone, C., R: Working with Databases, 2018, https://nuitrcs.github.io/databases_workshop/r/r_databases.html#execute-queries, accesat la 22.03.2019.

înregistrărilor se face după atributul nrfactura fiind afișate doar variabile specificate în funcția select().

Pentru ordonarea tuplurilor/observabilelor în rezultatul final în ordine crescătoare (ASC) sau descrescătoare (funcția desc()) folosim funcția arrange() din pachetul "dplyr". Aceasta ordonează observabilele în ordine descrescătoare după variabila specificata în cadrul funcției desc(). Valorile lipsă sunt sortate la finalul setului de date. De asemenea, R furnizează o suită standard de operatori: >, >=, <, <=, != (nu este egal), == (egal), ce permit selectarea observabilelor în funcție de un anumit criteriu stabilit în funcția filter().

Seturile de date pot conține sute sau chiar mii de variabile, prin urmare selectarea coloanelor care ne interesează se face cu funcția select(). Există o serie de funcții ce pot fi folosite în select(), precum: starts_with("abc") potrivește numele care începe cu "abc"; ends_with("abc") numele care se termină cu "abc"; contains("abc") cuvintele care conțin "abc"; matches("(.)\\1") selectează variabilele care se potrivesc cu expresia.

Funcția select() poate fi utilizat și pentru redenumirea variabilelor, însă elimină variabilele care nu sunt menționate explicit, fiind recomandate funcțiile rename() sau names() pentru redenumirea variabilelor. Funcția everything() este utilizată în cazul în care dorim să modificăm ordinea variabilelor fără a specifica expres fiecare variabilă ci doar cele de început.

Sintaxa la interogarea "*Să se obțină lista contractelor de închiriere încheiate în perioada 2017?*" a) ordonata crescător respectiv ordonată descrescător este:

SQL:	**R:**
SELECT seriesasiu as Masina, valoarecontract, garantie, (valoarecontract + garantie) AS VALOARE_CONTRACT FROM contract WHERE dataiesirevigoare >=DATE'2017-01-01' AND dataiesirevigoare <= DATE'2017-12-31' ORDER BY VALOARE_CONTRACT DESC; a)	lista_contracte <- contract %>% mutate(valoare_contract = valoarecontract + garantie) %>% filter(dataiesirevigoare>= '2017-01-01' & dataiesirevigoare <= '2017-12-31')%>% arrange(desc(valoare_contract))%>% select(seriesasiu, valoarecontract, garantie, valoare_contract)
SELECT seriesasiu as Masina, valoarecontract, garantie, (valoarecontract + garantie) AS VALOARE_CONTRACT FROM contract WHERE dataiesirevigoare >=DATE'2017-01-01' AND dataiesirevigoare <= DATE'2017-12-31' ORDER BY VALOARE_CONTRACT; b)	lista_contracte <- contract %>% mutate(valoare_contract = valoarecontract + garantie) %>% filter(dataiesirevigoare>= '2017-01-01' & dataiesirevigoare <= '2017-12-31')%>% arrange(valoare_contract)%>% select(seriesasiu, valoarecontract, garantie, valoare_contract)

Rezultatele interogărilor sunt prezentate în Anexa 2 Figura 2.

Atât în R cât și în SQL ordonarea valorilor unui atribut în ordine crescătoare nu necesită decât specificarea atributului în clauza ORDER BY/arrange(), în timp ce pentru a ordona descrescător valorile trebuie să folosim funcția desc() în R respectiv operatorul DESC în SQL. De asemenea, ambele limbaje de procesare permit ordonarea observabilelor în funcție de lista atributelor specificate în cadrul funcției arrange() respectiv clazua ORDER BY (după mai multe atribute), astfel:

SQL:	**R:**
SELECT seriesasiu as Masina, valoarecontract, garantie, (valoarecontract + garantie) AS VALOARE_CONTRACT FROM contract WHERE dataiesirevigoare >=DATE'2017-01-01' AND dataiesirevigoare <= DATE'2017-12-31' ORDER BY dataiesirevigoare, NRZILEINCHIRIERE, VALOARE_CONTRACT DESC;	lista_contracte_1 <- contract %>% mutate(valoare_contract = valoarecontract + garantie) %>% filter(dataiesirevigoare>= '2017-01-01' & dataiesirevigoare <= '2017-12-31')%>% arrange(dataiesirevigoare, nrzileinchiriere, desc(valoare_contract))%>% select(seriesasiu, valoarecontract, garantie, valoare_contract)

Rezultatul interogării este prezentat în Anexa 2 Figura 3.

Joncțiuni. Tipuri de joncțiuni

Fuzionare a două sau mai multe tabele este posibilă prin intermediul joncțiunilor. Prin urmare, în continuare vom face o scurtă prezentare a celor mai utilizate tipuri de joncțiuni și a rezultatelor puse în evidență de acestea.

Joncțiunea internă (INNER JOIN)

Dacă se respectă condiția de integritate referențială numărul înregistrărilor din tabela rezultată este egal cu numărul de înregistrări din tabela copil în cazul joncțiunilor interne. Cu alte cuvinte, când fiecare înregistrare din tabela părinte are cel puțin o înregistrare în tabela copil sau valorile cheii străine sunt nenule și se regăsesc în tabela părinte[79]. În cazul în care condiția de integritate referențială nu se respectă va trebui să folosim și alte tipuri de joncțiuni (RIGHT/LEFT/FULL JOIN) pentru a pune în evidență înregistrările ce nu au corespondent în tabela copil sau alte table întrucât joncțiunile se pot realiza între mai multe table nu doar între tabela părinte și tabela copil. Sintaxa interogării "*Să se calculeze/afișeze pentru fiecare factură valoarea chiriei pe zi a produselor, valoarea tva-ului aferent și valoarea contractului?*" este:

SQL:

[79] Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009, p. 152.

```
SELECT F.NRFACTURA, DATAINTRAREINVIGOARE, SUM(VALOAREFACTURA) AS
VALOARE_INCHIRIERI_PER_ZI, SUM(VALOARETVA) AS VALOARE_TVA_PER_ZI,
SUM(valoarefactura*nrzileinchiriere) AS VALOARE_TOTALA, C.IDCONTRACT, SUM(GARANTIE) AS
VALAORE_GARANTIE
FROM PRODUSE_INCHIRIATE PI LEFT JOIN DETALII_FACTURA DF ON PI.CODPROD = DF.CODPROD
INNER JOIN FACTURI F ON F.NRFACTURA = DF.NRFACTURA
INNER JOIN CONTRACT C ON C.IDCONTRACT = F.IDCONTRACT
GROUP BY 1,  C.IDCONTRACT;
```

R:

```
R153 <- produse_inchiriate %>%
inner_join(detalii_factura, by = "codprod")%>%
inner_join(facturi, by = "nrfactura")%>%
inner_join(contract, by = "idcontract")%>%
group_by(nrfactura, datafacturii, idcontract)%>%
summarise(VALOARE_INCHIRIERI_PER_ZI = sum(valoarefactura), VALOARE_TVA_PER_ZI =
sum(valoaretva), VALOARE_TOTALA = sum(valoarefactura*nrzileinchiriere), VALAORE_GARANTIE =
sum(garantie))%>%
select(nrfactura, datafacturii, VALOARE_INCHIRIERI_PER_ZI, VALOARE_TVA_PER_ZI,
VALOARE_TOTALA, VALAORE_GARANTIE)
R153
```

Rezultatul interogării este prezentat în Anexa 2 Figura 4.

Observăm că în acest caz obținem o tabelă ce conține doar înregistrările pentru care valoarea cheii străine este nenulă în tabela contracte, respectiv cele două tabele produse_inchiriate și contract nu se află intre-o relație părinte-copil.

În R funcția utilizată pentru a joncționa două sau mai multe tabele este inner_join() în cadrul căreia specificăm tabela și atributul de legătură dintre cele două tabele (by = “atribut de legătură”) similar cu clauza INNER JOIN din SQL.

Joncțiunea externă stânga (LEFT OUTER JOIN)

Pentru a pune în evidență produsele care nu au fost închiriate sau nu se regăsesc pe nici o factură folosim clauza LEFT JOIN din SQL respectiv funcția cu un comportament similar left_join() din R.

SQL:

```
SELECT C.IDCONTRACT, F.NRFACTURA, PI.denprod, DATAINTRAREINVIGOARE, SUM(pretunit) AS
VALOARE_INCHIRIERI_PER_ZI, SUM(tvaliniepr) AS VALOARE_TVA_PER_ZI,
SUM((PRETUNIT+TVALINIEPR)*NRZILEINCHIRIERE) AS VALOARE_TOTALA, C.IDCONTRACT
FROM PRODUSE_INCHIRIATE PI LEFT JOIN DETALII_FACTURA DF ON PI.CODPROD = DF.CODPROD
LEFT JOIN FACTURI F ON F.NRFACTURA = DF.NRFACTURA
LEFT JOIN CONTRACT C ON C.IDCONTRACT = F.IDCONTRACT
GROUP BY 1, 2, 3, C.IDCONTRACT
ORDER BY 2;
```

R:

```
R150 <- produse_inchiriate %>%
left_join(detalii_factura, by = "codprod")%>%
left_join(facturi, by = "nrfactura")%>%
left_join(contract, by = "idcontract")%>%
group_by(denprod, nrfactura, datafacturii)%>%
mutate(VALOARE_INCHIRIERI_PER_ZI = sum(pretunit), VALOARE_TVA_PER_ZI = sum(tvaliniepr),
VALOARE_TOTALA = sum((pretunit + tvaliniepr)*nrzileinchiriere))%>%
select(idcontract, nrfactura, datafacturii, denprod, VALOARE_INCHIRIERI_PER_ZI, VALOARE_TVA_PER_ZI,
VALOARE_TOTALA)%>%
arrange(nrfactura)
R150
```

Rezultatul interogării este prezentat în Anexa 2 Figura 5.

Observăm că singurul produs care nu apare pe nici una din facturi este produsul “Ochelari de condus night view” (Anexa 2 - Figura 5).

Joncțiunea externă la dreapta (RIGHT OUTER JOIN)

În ceea ce privește joncțiunea RIGHT JOIN rezultatul și sintaxa interogării este prezentată în Anexa 2 - Figura 6. Și în acest caz observăm că tabela rezultată nu conține același număr de înregistrări ca tabela rezultată în urma joncțiunii LEFT JOIN. Acest tip de joncțiune pune în evidență faptul că pentru un număr de șase contracte nu s-a emis încă nici o factură, însă nu mai apare și produsul care nu a fost încă închiriat de nici un client.

SQL:

```
SELECT C.IDCONTRACT, F.NRFACTURA, PI.denprod, DATAINTRAREINVIGOARE, SUM(pretunit) AS
VALOARE_INCHIRIERI_PER_ZI, SUM(tvaliniepr) AS VALOARE_TVA_PER_ZI,
SUM((PRETUNIT+TVALINIEPR)*NRZILEINCHIRIERE) AS VALOARE_TOTALA, C.IDCONTRACT
FROM PRODUSE_INCHIRIATE PI LEFT JOIN DETALII_FACTURA DF ON PI.CODPROD = DF.CODPROD
RIGHT JOIN FACTURI F ON F.NRFACTURA = DF.NRFACTURA
RIGHT JOIN CONTRACT C ON C.IDCONTRACT = F.IDCONTRACT
GROUP BY 1, 2, 3, C.IDCONTRACT
ORDER BY 2;
```

R:

```
R151 <- produse_inchiriate %>%
right_join(detalii_factura, by = "codprod")%>%
right_join(facturi, by = "nrfactura")%>%
right_join(contract, by = "idcontract")%>%
group_by(denprod, nrfactura, datafacturii)%>%
mutate(VALOARE_INCHIRIERI_PER_ZI = sum(pretunit), VALOARE_TVA_PER_ZI = sum(tvaliniepr),
VALOARE_TOTALA = sum((pretunit + tvaliniepr)*nrzileinchiriere))%>%
select(idcontract, nrfactura, datafacturii, denprod, VALOARE_INCHIRIERI_PER_ZI, VALOARE_TVA_PER_ZI,
VALOARE_TOTALA)%>%
arrange(nrfactura)
```

Rezultatul interogării este prezentat în Anexa 2 Figura 6.

Joncțiunea externă (FULL OUTER JOIN)

Mai jos prezentăm sintaxa sintaxa interogării unde utilizăm joncțiunea FULL JOIN din SQL și funcția full_join() din R pentru a pune în evidență toate înregistrările.

SQL:

```
SELECT C.IDCONTRACT, F.NRFACTURA, PI.denprod, DATAINTRAREINVIGOARE, SUM(pretunit) AS
VALOARE_INCHIRIERI_PER_ZI, SUM(tvaliniepr) AS VALOARE_TVA_PER_ZI,
SUM((PRETUNIT+TVALINIEPR)*NRZILEINCHIRIERE) AS VALOARE_TOTALA, C.IDCONTRACT
FROM PRODUSE_INCHIRIATE PI LEFT JOIN DETALII_FACTURA DF ON PI.CODPROD = DF.CODPROD
FULL JOIN FACTURI F ON F.NRFACTURA = DF.NRFACTURA
FULL JOIN CONTRACT C ON C.IDCONTRACT = F.IDCONTRACT
GROUP BY 1, 2, 3, C.IDCONTRACT
ORDER BY 2;
```

R:

```
R152 <- produse_inchiriate %>%
full_join(detalii_factura, by = "codprod")%>%
full_join(facturi, by = "nrfactura")%>%
full_join(contract, by = "idcontract")%>%
group_by(denprod, nrfactura, datafacturii)%>%
mutate(VALOARE_INCHIRIERI_PER_ZI = sum(pretunit), VALOARE_TVA_PER_ZI = sum(tvaliniepr),
VALOARE_TOTALA = sum((pretunit + tvaliniepr)*nrzileinchiriere))%>%
select(idcontract, nrfactura, datafacturii, denprod, VALOARE_INCHIRIERI_PER_ZI, VALOARE_TVA_PER_ZI,
VALOARE_TOTALA)%>%
arrange(nrfactura)
R152
```

Rezultatul interogării este prezentat în Anexa 2 Figura 7.

Spre deosebire de rezultatul obținut prin utilizarea joncțiunilor LEFT JOIN și RGHT JOIN care puneau în evidență anumite înregistrări, observăm că aici obținem atât produsele ce nu au fost închiriate cât și contractele pentru care nu s-a emis nici o factură (Figura 7 – Anexa 2).

3.3. Grupări

În R funcția group_by() din "dplyr" are un comportament similar ca și clauza GROUP BY din SQL. Astfel, tabela rezultată din tablele enumerate în clauza FROM (în SQL) și cele specificate în funcția inner_join (tabela, by = c(coloana_identică)) (în R) este împărțită în grupuri de înregistrări în funcție de valorile comune ale unui atribut sau variabile specificată în clauza/funcția GROUP BY/group_by(), cu deosebirea că în R variabilele de grupare sunt păstrate întotdeauna în rezultat, iar ordonarea se face mai întâi după variabila de grupare.

Grupări simple

Pentru a defini variabilele de agregare utilizăm funcția summarize() - permite obținerea unei singure valori pentru atributul după care se face gruparea. Sintaxa interogării scrisă în cele două limbaje de procesare a datelor la întrebarea "*Care este valoarea închirierii pe zi a produselor de pe fiecare factură în parte?*" este:

SQL:	**R:**
SELECT f.nrfactura, COALESCE(sum(valoarefactura+valoareTVA), 0) AS "TOTAL incasari cu TVA" FROM facturi f INNER JOIN detalii_factura df ON f.nrfactura = df.nrfactura WHERE f.nrfactura BETWEEN 100 AND 109 GROUP BY f.nrfactura ORDER BY f.nrfactura;	R97 <- facturi%>% inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>% group_by(nrfactura)%>% filter(nrfactura >= 100 & nrfactura <= 109)%>% summarise(Total_incasari_cu_TVA = sum(valoarefactura + valoaretva, na.rm = TRUE))%>% arrange(nrfactura)%>% select(nrfactura, Total_incasari_cu_TVA) R97
Rezultatul interogării este prezentat în Anexa 2 Figura 8.	

Pentru a obține rezultatul dorit utilizăm, pe lângă funcțiile group_by(), filter(), arrange() și select(), funcția summarize() pentru a obține o singură valoare a produselor închiriate prezente pe fiecare factură.

Gruparea după două sau mai multe criterii

La întrebarea "*Care este situaţia vânzărilor pe clienţi şi luni în anul 2016*?" sintaxa și rezultatul interogării în cele două limbaje sunt prezentate în Anexa 2 - Figura 9.

SQL:	**R:**
SELECT NUMECLIENT, EXTRACT(MONTH FROM DATAFACTURII), SUM(DISTINCT (VALOAREFACTURA + VALOARETVA)*C.NRZILEINCHIRIERE) AS VALOARE_TOTALA, COUNT(DISTINCT F.NRFACTURA) AS NR_FACTURI FROM CLIENTI CL INNER JOIN CONTRACT C ON CL.IDCLIENT = C.IDCLIENT INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD WHERE DATAFACTURII BETWEEN DATE'2016-01-01' AND DATE'2016-12-31' GROUP BY NUMECLIENT, EXTRACT(MONTH	R105 <-clienti%>% inner_join(contract, by = 'idclient')%>% inner_join(facturi, by = 'idcontract')%>% inner_join(detalii_factura, by = 'nrfactura')%>% inner_join(produse_inchiriate, by = 'codprod')%>% filter(datafacturii >= '2016-01-01' & datafacturii <='2016-12-31')%>% group_by(numeclient, month(datafacturii))%>% distinct(valoarefactura, valoaretva, nrzileinchiriere)%>% summarise(valoare_serviciu = sum((valoarefactura + valoaretva)*nrzileinchiriere), nr_facturi = n()) R105

```
FROM DATAFACTURII);
```

Rezultatul interogării este prezentat în Anexa 2 Figura 9.

Pentru a obține valoarea facturilor emise pe clienți și, pentru fiecare client, valoarea lunară a acesteia, vom face o grupare după atributele numeclient și lună.

Răspunsul la interogarea "*Care sunt facturile pentru care valoarea chiriei pe zi a produselor închiriate sunt mai mari de 6000 lei*" este obținut utilizând clauza HAVING care operează la nivel de grupuri de înregistrări și care nu are echivalent în R.

SQL:

```
SELECT f.nrfactura,
COALESCE(sum(valoarefactura+valoareTVA), 0) AS
"TOTAL incasari cu TVA"
FROM facturi f INNER JOIN detalii_factura df ON
f.nrfactura = df.nrfactura
WHERE f.nrfactura BETWEEN 100 AND 109
GROUP BY F.NRFACTURA
HAVING
COALESCE(sum(valoarefactura+valoareTVA), 0) >=
6000
ORDER BY f.nrfactura;
```

R:

```
R98 <- facturi%>%
inner_join(detalii_factura, by = 'nrfactura')%>%
group_by(nrfactura)%>%
summarise(Total_chirie_per_zi = sum(valoarefactura +
valoaretva, na.rm = TRUE))%>%
filter(Total_chirie_per_zi >= 6000)
R98
```

Rezultatul interogării este prezentat în Anexa 2 Figura 10.

Pachetul "dplyr" oferă un avantaj prin faptul că nu mai este necesar să rescriem expresia pentru filtrarea grupurilor de înregistrări ce au o valoare mai mare decât 6000 lei cum este cazul SQL-ului.

Un alt tip de consultări ce necesită utilizarea grupărilor este reprezentat de interogările ce includ subtotaluri și totaluri. Un exemplu în acest sens este "*Să se obțină situația vânzărilor pe fiecare birou de închiriere și an cu realizarea de subtotaluri pe fiecare birou?*" a căror sintaxă este:

SQL:

```
SELECT NUMEBIROU, EXTRACT(YEAR FROM DATAFACTURII), SUM(DISTINCT (PRETUNIT +
TVALINIEPR)*C.NRZILEINCHIRIERE) AS VALOARE_SERVICIU
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
GROUP BY NUMEBIROU, 2
UNION
SELECT NUMEBIROU || ' - SUBTOTAL', NULL,SUM( (PRETUNIT + TVALINIEPR)*C.NRZILEINCHIRIERE)
```

```
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
GROUP BY NUMEBIROU
ORDER BY 1, 2;
```

R:

```
R113 <- bind_rows(birouinchirieri%>%
                  inner_join(clasaauto, by = c('idbirouinchirieri', 'idbirouinchirieri'))%>%
                  inner_join(modeleautoturism, by = c('idclasa', 'idclasa'))%>%
                  inner_join(contract, by = c('seriesasiu', 'seriesasiu'))%>%
                  inner_join(facturi, by = c('idcontract', 'idcontract'))%>%
                  inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>%
                  inner_join(produse_inchiriate, by = c('codprod', 'codprod'))%>%
group_by( numebirou, year(datafacturii))%>%
distinct(pretunit, tvaliniepr, nrzileinchiriere, nrfactura)%>%
summarise(valoare_serviciu = sum((pretunit+tvaliniepr)*nrzileinchiriere))%>%
mutate(category = ifelse(numebirou %in% 'Birou 1', "1",
      ifelse(numebirou %in%  'Birou 2', "2",
        ifelse(numebirou == 'Birou 3',  "3"))) ) %>%
select(category, numebirou, everything()),
R110 %>%
select(category, numebirou, everything())%>% group_by(category) %>%
summarize(numebirou = "Total",
valoare_serviciu = sum(valoare_serviciu)))%>%
arrange(category, numebirou)
```

Rezultatul interogării este prezentat în Anexa 2 Figura 11.

În continuare prezentăm sintaxa interogării "*Să se obțină situația vânzărilor pe fiecare birou de închiriere și an cu realizarea de subtotaluri pe fiecare birou și un total la nivel de firmă?*" obținute în cele două limbaje de procesare a datelor:

SQL:

```
SELECT NUMEBIROU, EXTRACT(YEAR FROM DATAFACTURII), EXTRACT(MONTH FROM
DATAFACTURII), SUM( (PRETUNIT + TVALINIEPR)*C.NRZILEINCHIRIERE) AS VALOARE_SERVICIU
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
GROUP BY NUMEBIROU, 2, 3
UNION
SELECT NUMEBIROU || ' - SUBTOTAL', NULL, NULL, SUM((PRETUNIT +
TVALINIEPR)*C.NRZILEINCHIRIERE)
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
```

```
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
GROUP BY NUMEBIROU
UNION
SELECT 'TOTAL GENERAL', NULL, NULL, SUM((PRETUNIT + TVALINIEPR)*C.NRZILEINCHIRIERE)
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
ORDER BY 1, 2, 3;
```

R:

```
R115 <- bind_rows(birouinchirieri%>%
                  inner_join(clasaauto, by = c('idbirouinchirieri', 'idbirouinchirieri'))%>%
                  inner_join(modeleautoturism, by = c('idclasa', 'idclasa'))%>%
                  inner_join(contract, by = c('seriesasiu', 'seriesasiu'))%>%
                  inner_join(facturi, by = c('idcontract', 'idcontract'))%>%
                  inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>%
                  inner_join(produse_inchiriate, by = c('codprod', 'codprod'))%>%
                  group_by( numebirou, year(datafacturii), month(datafacturii))%>%
distinct(pretunit, tvaliniepr, nrzileinchiriere, nrfactura)%>%
summarise(valoare_serviciu = sum((pretunit+tvaliniepr)*nrzileinchiriere))%>%
mutate(category = ifelse(numebirou %in% 'Birou 1', "1",
  ifelse(numebirou %in%  'Birou 2', "2",
    ifelse(numebirou == 'Birou 3',  "3")))) %>%
select(category, numebirou, everything()),
R110 %>%
select(category, numebirou, everything())%>%
group_by(category) %>%
summarize(numebirou = "Total",
valoare_serviciu = sum(valoare_serviciu)))%>%
arrange(category, numebirou) %>%
bind_rows(birouinchirieri%>%
          inner_join(clasaauto, by = c('idbirouinchirieri', 'idbirouinchirieri'))%>%
          inner_join(modeleautoturism, by = c('idclasa', 'idclasa'))%>%
          inner_join(contract, by = c('seriesasiu', 'seriesasiu'))%>%
          inner_join(facturi, by = c('idcontract', 'idcontract'))%>%
          inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>%
          inner_join(produse_inchiriate, by = c('codprod', 'codprod'))%>%
summarise(category = "",
              numebirou = "GrandTotal",
              valoare_serviciu = sum((pretunit+tvaliniepr)*nrzileinchiriere)))
```

Rezultatul interogării este prezentat în Anexa 2 Figura 12.

Pentru realizarea de subtotaluri și totaluri am utilizat funcția bind_rows() care este echivalentă cu clauza UNION din SQL. Se poate observa că atunci când grupăm după mai multe

variabile, avem câte un total pentru fiecare nivel de grupare în parte, iar pentru a îndepărta gruparea putem utiliza funcția ungroup(). De asemenea, funcția ungroup() este recomandată a fi utilizată după aplicarea funcțiilor group_by() și summarizare(). O altă funcție ce are un comportament similar cu operatorul UNION este rbind(). De asemenea, se pot obține totaluri și subtotaluri și pe linii folosind funcțiile cbind() (din pachetele de bază) și bind_cols() (din pachetul dplyr) din R.

3.4. Subinterogări

Spre deosebire de R, limbajul SQL permite plasarea de subinterogări atât în clauza WHERE, cât și în clauzele HAVING, FROM și SELECT. În continuare prezentăm câteva exemple în acest sens.

3.4.1. Subinterogări în clauza WHERE

În Anexa 2 - Figura 13 este prezentata soluția la întrebarea "*Care sunt facturile emise în aceeași zi cu factura numărul 109?*".

SQL:	**R:**
SELECT NRFACTURA FROM FACTURI WHERE DATAFACTURII IN (SELECT DATAFACTURII FROM FACTURI WHERE NRFACTURA = 109);	R122 <- facturi%>% filter(datafacturii %in% (facturi%>% filter(nrfactura == 109))[['datafacturii']])%>% select(nrfactura) R122

Rezultatul interogării este prezentat în Anexa 2 Figura 13.

În rezultatul interogării este inclusă și factura cu numărul 109. În acest caz am utilizat și operatorul %in% care are un comportament similar cu operatorul IN din SQL. Dacă dorim să excludem din rezultat factura de referință sintaxa este următoarea:

SQL:	**R:**
SELECT NRFACTURA FROM FACTURI WHERE NRFACTURA IN (SELECT NRFACTURA FROM FACTURI WHERE NRFACTURA = 109) AND NRFACTURA <> 109	R123 <- facturi%>% filter(datafacturii %in% (facturi%>% filter(nrfactura == 109))[['datafacturii']])%>% select(nrfactura)%>% filter(nrfactura != 109) R123

După cum se poate observa în R am folosit o subinterogare în funcția filter() pentru a selecta doar data calendaristică ((facturi1%>% filter(nrfactura == 109))[['datafacturii']]) în care a fost emisă factura cu numărul 109 pentru a filtra doar facturile emise pe aceeași dată cu aceasta și care ar putea fi considerată ca fiind echivalentă cu subinterogarea din clauza WHERE din SQL.

3.4.2. Subinterogări în clauza HAVING

La întrebarea "*Care este anul în care au fost emise cele mai multe facturi?*" sintaxa și rezultatul obținut sunt prezentate în Anexa 2 - Figura 14. Subconsulatarea din clauza HAVING calculează numărul de facturi emise în fiecare an (GROUP BY EXTRACT(YEAR FROM datafacturii)), în cazul nostru se obțin trei valori corespunzători celor trei ani, care este comparată cu numărul de facturi corespunzător fiecărui an cu toate valorile extrase din subconsultare.

SQL:	**R:**
SELECT EXTRACT(YEAR FROM DATAFACTURII), COUNT(*) FROM FACTURI GROUP BY 1 HAVING COUNT(*) >= ALL(SELECT COUNT(*) FROM FACTURI GROUP BY EXTRACT(YEAR FROM DATAFACTURII));	R134<- facturi%>% group_by(year(datafacturii))%>% tally()%>% filter(n>= max(n)) R134
Rezultatul interogării este prezentat în Anexa 2 Figura 14.	

În ceea ce privește sintaxa interogării scrise în R, observăm că este ceva mai elegantă și mai suplă. Aici am folosit funcția tally() împreună cu funcția group_by() care este similară cu funcția n() utilizată pentru numărarea facturilor din fiecare an. Este, de asemenea, similară și cu funcția count() care apelează înainte de a număra funcția group_by() și la final funcția ungroup()[80].

3.4.3. Subinterogări în clauza FROM

Clauza FROM ne permite să definim tabele ad-hoc, nu doar să specificăm tabelele bazei de date. Pentru întrebarea "*Care este valoarea salariilor și valoarea salarială medie a angajaților Biroului 3?*" am formulat următoarea soluție a cărei sintaxă și rezultat este prezentat în Anexa 2 - Figura 15.

SQL:	**R:**
SELECT * FROM (SELECT SUM(SALAR) AS TOTAL_SALARII FROM ANGAJATI A INNER JOIN BIROUINCHIRIERI B ON A.IDBIROUINCHIRIERI = B.IDBIROUINCHIRIERI WHERE NUMEBIROU = 'Birou 3') AS TOTAL, (SELECT ROUND(AVG(SALAR), 2) MEDIA_SALARIALA FROM ANGAJATI A INNER JOIN BIROUINCHIRIERI B ON	R139<- birouinchirieri%>% inner_join(angajati, by = 'idbirouinchirieri')%>% filter(numebirou == "Birou 3")%>% summarise(total_salarii = sum((salar), na.rm = TRUE), media_salariala = mean(salar, na.rm = TRUE))%>% select(total_salarii, media_salariala) R139

[80] ***Count/tally observations by group, https://dplyr.tidyverse.org/reference/tally.html,accesat la 30.04.2019.

```
A.IDBIROUINCHIRIERI = B.IDBIROUINCHIRIERI
WHERE NUMEBIROU = 'Birou 3') AS TOTAL2;
```

Rezultatul interogării este prezentat în Anexa 2 Figura 15.

Soluția formulată include în clauza FROM două subinterogari scalare. În ceea ce privește limbajul R observăm că sintaxa soluției prezentate este mult mai scurtă și mai elegantă.

În Anexa 2 - Figura 16 este prezentată sintaxa și rezultatul obținut pentru interogarea "*Care este prețul mediu de închiriere a mașinilor pe ani și luni?*" în cadrul căreia este pusă în evidență subinterogarea din cadrul funcției inner_join() similar cu cea folosita în sintaxa SQL. Pachetul lubridate pune la dispoziția utilizatorilor funcții dedicate pentru extragerea anului și lunii din data calendaristică precum year() și month(). De asemena, pentru calcularea prețului mediu s-a utilizat funcția mean() din R și AVG() din SQL.

SQL:

```
SELECT EXTRACT(YEAR FROM DATAFACTURII)
as AN, EXTRACT(MONTH FROM DATAFACTURII)
as LUNA,
COUNT(DISTINCT P.CODPROD) as NR_PRODUS,
ROUND(AVG(VALOAREFACTURA +
VALOARETVA), 2) as avgprice, SUM(liniefactura) as
NR_UNITATI FROM FACTURI F JOIN
DETALII_FACTURA DF ON F.NRFACTURA =
DF.NRFACTURA JOIN  (SELECT * FROM
PRODUSE_INCHIRIATE
WHERE UPPER(DENPROD) =
UPPER('AUTOTURISM') ) p
ON DF.CODPROD = p.CODPROD
GROUP BY EXTRACT(YEAR FROM
DATAFACTURII), EXTRACT(MONTH FROM
DATAFACTURII)
ORDER BY 1, 2;
```

R:

```
R38 <- facturi %>%
inner_join(detalii_factura, by = c('nrfactura',
'nrfactura'))%>%
inner_join((produse_inchiriate %>%
            filter(toupper(denprod) ==
toupper('autoturism'))), by = c('codprod', 'codprod'))%>%
group_by(year(datafacturii), month(datafacturii))%>%
mutate(avgprice = round(mean(valoarefactura +
valoaretva)), NR_UNITATI = sum(liniefactura))%>%
select(avgprice, NR_UNITATI)
R38
```

Rezultatul interogării este prezentat în Anexa 2 Figura 16.

3.4.4. Subinterogări în clauza SELECT

Spre deosebire de clauzele FROM, WHERE și HAVING unde puteam include subinterogări care aveau ca rezultat obținerea unei tabele, în cadrul clauzei SELECT putem include doar subinterogări scalare[81].

Soluția găsită la interogarea "*Care sunt vânzările totale și care este valoarea medie a vânzărilor pe contract?*" are următoarea sintaxă:

[81] Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009, p. 428.

SQL:	R:
SELECT ROUND((SELECT SUM(VALOARECONTRACT) FROM CONTRACT),2) AS VANZARI, ROUND((SELECT AVG(VALOARECONTRACT) FROM CONTRACT WHERE VALOARECONTRACT <> 0), 2) AS MEDIA_VANZARILOR_CONTRACT;	R138<- contract%>% summarise(vanzari = sum((valoarecontract), na.rm = TRUE), media_vanzarilor_contract = mean(valoarecontract, na.rm = TRUE))%>% select(vanzari, media_vanzarilor_contract) R138

Rezultatul interogării este prezentat în Anexa 2 Figura 17.

În cadrul frazei SELECT am inclus două subconsultări scalare, una care extrage vânzările totale la nivelul întregii organizații, și una care calculează valoarea medie a vânzărilor pe contract. Observăm că, spre deosebire de alte SGBD-uri, în PostgreSQL clauza FROM poate lipsi. În R soluția prezentata folosește doar funcția summarize() pentru calculul valorii totale a vânzărilor și a valorii medii și funcția select() pentru selectarea variabilelor.

3.5. Diferențe, reuniuni și intersecții

În R funcția echivalentă operatorului EXCEPT este setdiff() utilizată în obținerea răspunsului la interogarea "*Care sunt produsele care apar pe factura numărul 100 și nu apar pe factura numărul 108?*":

SQL:

```
SELECT DENPROD
FROM FACTURI F INNER JOIN DETALII_FACTURA DF ON
DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD =
DF.CODPROD
WHERE F.NRFACTURA = 108
EXCEPT
SELECT DENPROD
FROM FACTURI F INNER JOIN DETALII_FACTURA DF ON
DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD =
DF.CODPROD
WHERE F.NRFACTURA = 100;
```

R:

```
R126<- dplyr::setdiff(
facturi%>%
inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>%
inner_join(produse_inchiriate, by = c('codprod', 'codprod'))%>%
filter(nrfactura == 108)%>%
select(denprod),
facturi%>%
inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>%
inner_join(produse_inchiriate, by = c('codprod', 'codprod'))%>%
```

```
filter(nrfactura == 100)%>%
select(denprod))
R126
```

Rezultatul interogării este prezentat în Anexa 2 Figura 18.

Rezultatul interogării arată că pe factura numărul 100 nu se regăsește produsul "Căruț copil" aflat pe factura numărul 108. În timp ce, în cazul în care inversăm cele doua interogări, pe factura numărul 108 nu se regăsește produsele GPS și Scaun copil Auto. Înregistrările duplicat sunt automat eliminate atunci când utilizăm operatorul EXCEPT din SQL respectiv setdiff() din R, iar pentru includerea lor în SQL adăugăm clauza ALL, în R nu am găsit echivalent.

Pentru a vedea "*Care sunt produsele comune de pe cele doua facturi (100 și 108)*" apelăm la operatorul INTERSECT din SQL și funcția intersect() din R care va include în rezultat doar înregistrările comune celor două interogări:

SQL:

```
SELECT DENPROD
FROM FACTURI F INNER JOIN
DETALII_FACTURA DF ON DF.NRFACTURA =
F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON
PI.CODPROD = DF.CODPROD
WHERE F.NRFACTURA = 100
INTERSECT
SELECT DENPROD
FROM FACTURI F INNER JOIN
DETALII_FACTURA DF ON DF.NRFACTURA =
F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON
PI.CODPROD = DF.CODPROD
WHERE F.NRFACTURA = 108;
```

R:

```
R129<- dplyr::intersect(
facturi%>%
inner_join(detalii_factura, by = "nrfactura")%>%
inner_join(produse_inchiriate, by = "codprod")%>%
filter(nrfactura == 108)%>%
select(denprod),
facturi%>%
inner_join(detalii_factura, by = "nrfactura")%>%
inner_join(produse_inchiriate, by ="codprod")%>%
filter(nrfactura == 100)%>%
select(denprod))
R129
```

Rezultatul interogării este prezentat în Anexa 2 Figura 19.

Rezultatul interogării evidențiază produsele comune regăsite pe cele două facturi, "Autoturism" respectiv "Plasă separatoare animal". Pentru a include în rezultat și observabilele duplicat folosim funcția intersect_all(). Standardul SQL nu permite prezența înregistrărilor duplicat în rezultat decât în cazul în care condiția de unicitate nu este respectată. În PostgreSQL operatorul INTERSECT ALL furnizează același rezultat ca și operatorul INTERSECT[82].

În ceea ce privește interogarea "*Să se obțină situația vânzărilor pe birou și pe ani cu un subtotal pe fiecare birou și un total general la final?* " sintaxa soluției găsite în cele două limbaje este:

[82] Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009, p. 183-184.

SQL:

```
SELECT NUMEBIROU, EXTRACT(YEAR FROM DATAFACTURII), SUM(DISTINCT (PRETUNIT +
TVALINIEPR)*C.NRZILEINCHIRIERE) AS VALOARE_SERVICIU
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
GROUP BY NUMEBIROU, 2
UNION
SELECT NUMEBIROU || ' - SUBTOTAL', NULL,SUM(DISTINCT (PRETUNIT +
TVALINIEPR)*C.NRZILEINCHIRIERE)
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
GROUP BY NUMEBIROU
UNION
SELECT 'TOTAL GENERAL', NULL,  SUM(DISTINCT (PRETUNIT + TVALINIEPR)*C.NRZILEINCHIRIERE)
FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO CA ON B.IDBIROUINCHIRIERI =
CA.IDBIROUINCHIRIERI
INNER JOIN MODELEAUTOTURISM MA ON CA.IDCLASA = MA.IDCLASA
INNER JOIN CONTRACT C ON MA.SERIESASIU = C.SERIESASIU
INNER JOIN FACTURI F ON C.IDCONTRACT = F.IDCONTRACT
INNER JOIN DETALII_FACTURA DF ON DF.NRFACTURA = F.NRFACTURA
INNER JOIN PRODUSE_INCHIRIATE PI ON PI.CODPROD = DF.CODPROD
ORDER BY 1, 2;
```

R:

```
R114 <- bind_rows(birouinchirieri%>%
                  inner_join(clasaauto, by = c('idbirouinchirieri', 'idbirouinchirieri'))%>%
                  inner_join(modeleautoturism, by = c('idclasa', 'idclasa'))%>%
                  inner_join(contract, by = c('seriesasiu', 'seriesasiu'))%>%
                  inner_join(facturi, by = c('idcontract', 'idcontract'))%>%
                  inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>%
                  inner_join(produse_inchiriate, by = c('codprod', 'codprod'))%>%
                  group_by( numebirou, year(datafacturii))%>%
                  distinct(pretunit, tvaliniepr, nrzileinchiriere, nrfactura)%>%
                  summarise(valoare_serviciu = sum((pretunit+tvaliniepr)*nrzileinchiriere))%>%

                  mutate(category = ifelse(numebirou %in% 'Birou 1', "1",
                                    ifelse(numebirou %in%  'Birou 2', "2",
                                           ifelse(numebirou == 'Birou 3',  "3")))
                  ) %>%
                  select(category, numebirou, everything()),
                 R110 %>%
                  select(category, numebirou, everything())%>%
                  group_by(category) %>%
                  summarize(numebirou = "Total",
                            valoare_serviciu = sum(valoare_serviciu)))%>%
```

```
 arrange(category, numebirou) %>%
 bind_rows(birouinchirieri%>%
        inner_join(clasaauto, by = c('idbirouinchirieri', 'idbirouinchirieri'))%>%
        inner_join(modeleautoturism, by = c('idclasa', 'idclasa'))%>%
        inner_join(contract, by = c('seriesasiu', 'seriesasiu'))%>%
        inner_join(facturi, by = c('idcontract', 'idcontract'))%>%
        inner_join(detalii_factura, by = c('nrfactura', 'nrfactura'))%>%
        inner_join(produse_inchiriate, by = c('codprod', 'codprod'))%>%
        # select(numebirou, pretunit, tvaliniepr, nrzileinchiriere, datafacturii)%>%
        summarise( category = "",
                numebirou = "GrandTotal",
                valoare_serviciu = sum((pretunit+tvaliniepr)*nrzileinchiriere)))
```

R114

Rezultatul interogării este prezentat în Anexa 2 Figura 20.

Soluția formulată folosește pentru reunirea înregistrărilor obținute din fiecare interogare funcția union() în R ce are un comportament identic cu funcția bind_rows() (respectiv rbind() din pachetul de bază) și care sunt similare cu clauza UNION din SQL. La reuniune SQL, ca și R, elimină din rezultat înregistrările duplicat. În cazul în care dorim să includem în rezultat și înregistrările duplicat folosim clauza ALL în SQL și funcția union_all() în R.

3.6. Tabele pivot

O caracteristică esențială a sistemelor de tip BI în ceea ce privește raportarea și analiza datelor este dată de utilizarea tabelelor pivot. În PostgreSQL funcția CROSSTAB necesită definirea explicită a coloanelor din operația de pivotare[83], devenind o problemă în cazul în care avem de definit un număr mare de coloane. Sintaxa interogării ce permite afișarea vânzărilor pe zi a produselor închiriate pe perioada 2016-2019 este prezentată în continuare:

SQL:

```
WITH REZULTAT3 AS (
select * from crosstab (
SELECT DISTINCT AN, LUNA, COALESCE(VALOARE, 0) AS valoare_serviciu
FROM raport3
GROUP BY AN, LUNA
ORDER BY AN',
'SELECT DISTINCT LUNA FROM  raport3 ORDER BY 1' )
 AS (AN TEXT , "Februarie" NUMERIC,  "Mai" NUMERIC, "Iunie" NUMERIC, "Iulie" NUMERIC, "August"
NUMERIC, "Septembrie" NUMERIC, "Octombrie" NUMERIC, "Noiembrie" NUMERIC ) ),
TOTAL AS (select AN::TEXT, SUM(VALOARE) AS VANZARI_ZI
FROM RAPORT3
GROUP BY AN)
SELECT  R.*, T.VANZARI_ZI
FROM REZULTAT3 R INNER JOIN TOTAL T ON T.AN =  R.AN;
```

[83] https://www.postgresql.org/docs/9.1/tablefunc.html, accesat la 30.04.2019.

R:

```
R174<-cbind(data.frame(facturi %>%
                       inner_join(detalii_factura, by = "nrfactura")%>%
                       group_by(year = year(datafacturii), month = month(datafacturii)) %>%
                       arrange(year, month) %>%
                       summarise(vanz = sum(pretunit+tvaliniepr)) %>%
                       spread(month, vanz, fill = 0)%>%
                       select(anul = year, "ianuarie" = 2, "februarie" = 3, "mai" = 4, "iunie" = 5,  "iulie" = 6,  "august" = 7,
"septembrie" = 8, "noiembrie" = 9)

)  , facturi %>%
  inner_join(detalii_factura, by = "nrfactura")%>%
  group_by(year = year(datafacturii)) %>%
  arrange(year) %>%
  summarise(vanz = sum(coalesce(pretunit, 0) + coalesce(tvaliniepr, 0))) %>%
  select(vanz))
R174
```

Rezultatul interogării este prezentat în Anexa 2 Figura 21.

În continuare prezentăm sintaxa la întrebarea "*Care sunt vânzările pe zi a produselor închiriate în perioada 2016 - 2019?*". În acest caz am folosit funcția CROSSTAB() din SQL pentru realizarea tabelelor pivot care este echivalenta cu funcția spread() din R. Pentru a putea folosi funcția CROSSTAB() trebuie activat modulul "tablefunc" astfel: CREATE EXTENSION tablefunc;[84].

Funcția CROSSTAB() primește o comandă SELECT ca parametru care trebuie să satisfacă următoarele restricții: fraza SELECT trebuie să returneze cel puțin trei coloane; prima coloană joacă rol de identificator a rândurilor din tabela pivot în cazul nostru este coloana AN, a doua coloană reprezintă categoriile din tabela pivot în cazul de fata sunt lunile din an, iar a treia coloană este valoarea corespunzătoare fiecărei celule (vanzari_zi). Pentru a obține valoarea totală pe linie/an am folosit o interogare de tip expresii-tabelă. În acest caz am definit două expresii-tabelă REZULTAT3 (tabelă obținută cu ajutorul funcției CROSSTAB) și TOTAL (tabelă obținută print-o frază SELECT simplă). Fraza SELECT principală nu face altceva decât să unească coloanele celor două tabele ad-hoc obținute. În R rezultatul final a fost obținut tot prin intermediul a două interogări unite cu ajutorul funcție cbind(). Prima interogare folosește funcția spread() pentru a așeza pe coloane lunile și valorile vânzărilor pe zi corespunzătoare fiecărei celule. A doua interogare permite obținerea totalului pe linie/an.

[84] ***Metrics Maven: Creating Pivot Tables in PostgreSQL Using Crosstab, 2016, https://www.compose.com/articles/metrics-maven-creating-pivot-tables-in-postgresql-using-crosstab/, accesat la 07.05.2019.

3.7. Recursivitate

Soluția găsită la interogarea "*Sa se afișeze mărcile de mașini deținute de fiecare birou?*" returnează toate mărcile de mașini deținute de fiecare birou în parte sub forma unui șir de caractere. Funcția utilizată în SQL pentru concatenarea modelelor de mașini este string_agg() ce poate fi considerată similară cu funcția paste() din R.

SQL:	**R:**
SELECT B.NUMEBIROU, STRING_AGG(MARCAAUTO||' ' || MODEL,' , ') FROM BIROUINCHIRIERI B INNER JOIN CLASAAUTO C ON C.IDBIROUINCHIRIERI=B.IDBIROUINCHIRIERI INNER JOIN MODELEAUTOTURISM MA ON MA.IDCLASA=C.IDCLASA GROUP BY 1 ORDER BY 1;	R48 <- birouinchirieri%>% inner_join(clasaauto, by = "idbirouinchirieri")%>% inner_join(modeleautoturism, by = "idclasa")%>% group_by(idbirouinchirieri, numebirou)%>% summarise(marci_detinute = paste(marcaauto, collapse = ', '))%>% select(idbirouinchirieri, numebirou, marci_detinute) R48

Rezultatul interogării este prezentat în Anexa 2 Figura 22.

3.8. Expresii-tabelă

Prezentăm în continuare sintaxa soluției bazată pe expresii-tabelă obținută în SQL și soluția obținută în R pentru afișarea "*raportului privind vânzările din perioada 2016 - 2019, pe produs cu total pe linii și coloane*".

SQL:

```
WITH raportvanzari("codprod", "denprod", "An", "Ian", "Febr", "Mar", "Apr", "Mai", "Iun", "Iul", "Aug", "Sep",
"Oct", "Nov", "Dec")
AS ( SELECT pi.codprod, denprod, (EXTRACT(YEAR FROM datafacturii))::TEXT,
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=1 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END) AS "Ian",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=2 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Febr",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=3 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Mar",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=4 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Apr",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=5 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Mai",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=6 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Iun",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=7 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Iul",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=8 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Aug",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=9 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Sep",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=10 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE
```

```
0 END)AS "Oct",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=11 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE
0 END)AS "Nov",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=12 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE
0 END)AS "Dec"
FROM facturi f INNER JOIN detalii_factura df ON df.nrfactura = f.nrfactura
INNER JOIN produse_inchiriate pi ON pi.codprod = df.codprod
INNER JOIN CONTRACT C ON C.IDCONTRACT = F.IDCONTRACT
GROUP BY 1, 2, 3
ORDER BY 2)
SELECT "denprod", COALESCE("An", 'Total')  AS "An", sum("Ian") AS "Ian", sum("Febr") AS "Febr",
sum("Mar") AS "Mar", sum("Apr") AS "Apr", sum("Mai") AS "Mai", sum("Iun") AS "Iun", sum("Iul") AS "Iul",
sum("Aug") AS "Aug", sum("Sep") AS "Sep", sum("Oct") AS "Oct", sum("Nov") AS "Nov", sum("Dec") AS "Dec",
"Ian"+"Febr"+"Mar"+"Apr"+"Mai"+"Iun"+"Iul"+"Aug"+"Sep"+"Oct"+"Nov"+"Dec" AS "TotalAn"
FROM raportvanzari
GROUP BY 1, 2, 15
UNION
SELECT coalesce(null, 'Total'), coalesce(null, ''),
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=1 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END) AS "Ian",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=2 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Febr",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=3 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Mar",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=4 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Apr",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=5 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Mai",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=6 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Iun",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=7 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Iul",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=8 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Aug",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=9 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE 0
END)AS "Sep",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=10 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE
0 END)AS "Oct",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=11 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE
0 END)AS "Nov",
sum(CASE WHEN EXTRACT(MONTH FROM datafacturii)=12 THEN (pretunit+tvaliniepr)*nrzileinchiriere ELSE
0 END)AS "Dec", coalesce(null, (select sum((pretunit + tvaliniepr)*nrzileinchiriere)
FROM facturi f INNER JOIN detalii_factura df ON df.nrfactura = f.nrfactura
INNER JOIN produse_inchiriate pi ON pi.codprod = df.codprod
INNER JOIN CONTRACT C ON C.IDCONTRACT = F.IDCONTRACT))
FROM facturi f INNER JOIN detalii_factura df ON df.nrfactura = f.nrfactura
INNER JOIN produse_inchiriate pi ON pi.codprod = df.codprod
INNER JOIN CONTRACT C ON C.IDCONTRACT = F.IDCONTRACT
ORDER BY 1;
```

R:

```
R88 <- cbind(contract %>%
             inner_join(facturi, by = "idcontract")%>%
             inner_join(detalii_factura,by = "nrfactura")%>%
             inner_join(produse_inchiriate, by ="codprod")%>%
             group_by(denprod, year = year(datafacturii),month = month(datafacturii)) %>%
```

```
            arrange(denprod, year, month) %>%
            summarise(vanz = sum((pretunit+tvaliniepr)*nrzileinchiriere)) %>%
            spread(month, vanz, fill = 0)%>%
            select(anul = 2, "ianuarie" = 3, "februarie" = 4, "mai" = 5, "iunie" = 6,  "iulie" = 7,  "august" = 8,
"septembrie" = 9, "noiembrie" = 10)
          ,  contract %>%
            inner_join(facturi, by = "idcontract")%>%
            inner_join(detalii_factura,by = "nrfactura")%>%
            inner_join(produse_inchiriate, by ="codprod")%>%
            group_by(denprod, year = year(datafacturii)) %>%
            arrange(denprod) %>%
            dplyr::summarise(vanzari_2016_2019 = sum(coalesce((pretunit+tvaliniepr)*nrzileinchiriere, 0) )) %>%
            select(vanzari_2016_2019))%>%
 bind_rows(
     contract %>%
     inner_join(facturi, by = "idcontract")%>%
     inner_join(detalii_factura,by = "nrfactura")%>%
     group_by(month = month(datafacturii)) %>%
     arrange(month) %>%
     dplyr::summarise(denprod = 'Total',anul = NA,
     vanz = sum(coalesce((pretunit)*nrzileinchiriere, 0) + coalesce(tvaliniepr*nrzileinchiriere, 0))) %>%
     spread(month, vanz, fill = 0)%>%
     mutate(vanzari_2016_2019 = (contract %>%
                                 inner_join(facturi, by = "idcontract")%>%
                                 inner_join(detalii_factura,by = "nrfactura")%>%
                                 summarise(vanz = sum((pretunit+tvaliniepr)*nrzileinchiriere)))[['vanz']])%>%
     select(denprod = 1, anul = 2, "ianuarie" = 3, "februarie" = 4, "mai" = 5, "iunie" = 6,  "iulie" = 7,  "august" = 8,
"septembrie" = 9, "noiembrie" = 10, vanzari_2016_2019))
R88
```

Rezultatul interogării este prezentat în Anexa 2 Figura 23.

În acest caz pentru a obține totalurile pe luni și pe produs în perioada 2016 - 2019 am folosit o soluție bazată pe expresii-tabelă în care am creat tabela raportvanzari ce conține vânzările pe fiecare produs, din fiecare lună. Fraza SELECT principală selectează coloanele ce ne interesează în rezultatul final (denprod, an etc.) și adaugă coloana ce realizează totalul pe fiecare produs. Pentru a obține totalurile pe fiecare lună în parte am apelat la o altă interogare ce a fost adăugată la prima interogare cu ajutorul operatorului UNION.

În ceea ce privește soluția găsită în R, aici am folosit o combinație de funcții precum cbind() pentru a adăuga totalurile pe fiecare produs și an, respectiv bind_rows() pentru a alipi la prima interogare totalurile pe fiecare coloană similare cu UNION din SQL. În cadrul funcției cbind() am folosit funcția spread() cu un comportament similar ca funcția CROSSTAB() din SQL.

3.9. Funcții OLAP

Funcțiile OLAP sunt cunoscute și sub numele de funcții analitice și au fost adăugate în standardul SQL1999. Pachetul "tidyverse" oferă o serie de funcții analitice precum: lead(), lag(), dense_rank(), min_rank(), procent_rank(), row_number(), ntile(), cume_dist() - distribuția cumulativă, cumall(), top_n() și care funcționează similar ca cele din SQL.

Un exemplu de interogare în care utilizăm o funcție analitică este "*Să se afișeze pentru fiecare departament, valoarea medie a salariilor, și salariile fiecărui angajat.*", a cărei sintaxă, scrisă în limbajul SQL și R, este:

SQL:	**R:**
SELECT NUMEBIROU, NUMEDEPART, ROUND(AVG(SALAR) OVER (PARTITION BY NUMEDEPART), 2) FROM ANGAJATI A INNER JOIN BIROUINCHIRIERI B ON A.IDBIROUINCHIRIERI = B.IDBIROUINCHIRIERI INNER JOIN DEPARTAMENTE D ON D.IDDEPART = A.IDDEPART GROUP BY 1, 2, A.SALAR;	R143<- birouinchirieri%>% inner_join(angajati, by = c('idbirouinchirieri', 'idbirouinchirieri'))%>% inner_join(departamente, by = c('iddepart', 'iddepart'))%>% merge(aggregate(salar ~ numedepart, angajati %>% inner_join(departamente, by = "iddepart"), mean), by = "numedepart")%>% select(numebirou, numedepart, salar.x, salar.y) R143
Rezultatul interogării este prezentat în Anexa 2 Figura 24.	

Adăugarea la funcția agregat AVG() a operatorului OVER face ca funcția să devină o funcție fereastră. Partiția are dimensiunea unui departament, iar numărul de înregistrări ce alcătuiesc partiția depinde de numărul de angajați ce fac parte din acel departament. În cadrul partiției ordonarea se face după numărul facturii. Calculul se efectuează pentru fiecare înregistrare din cadrul ferestrei, având, în acest caz, o margine fixă dată de începutul partiției și una mobilă dată de linia curentă/finalul partiției. În cadrul unei ferestre putem utiliza orice funcție de agregare. Rezultatul interogării este valoarea medie a salariului, după fiecare salariat, pentru fiecare departament.

Capitolul IV Măsurarea timpului de execuție a interogarilor în R și SQL

Testele au fost realizate pe un calculator pe care este instalat sistemul de operare Windows 10 Pro 64 bit, cu un procesor Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz și cu o memorie RAM de 8GB.

Pentru crearea bazei de date "Închirieri Mașini" s-a folosit un sistem de gestiune a bazelor de date versiunea PgAdmin Portable 2.2.0.0 (PostgreSQL 9.6). În ceea ce privește R-ul versiunea utilizată este R 3.4.0 (2017-04-21).

Au fost testate diferite tipuri de interogări DML (în principal fraze SELECT) pentru a analiza timpul de execuție (performanta tehnologiilor) a interogărilor redactate în cele două limbaje de procesare a datelor SQL și R.

Tabelul 2. Numărul de înregistrări pentru fiecare tabele și dimensiunea bazei de date.

Nume tabel	**Dimensiunea bazei de date**						
	9MB	11MB	15MB	36MB	82MB	94MB	175MB
angajati	18	100	150	200	200	300	300
modeleautoturism	22	101	130	200	160	200	201
clineti	19	1001	1296	2001	1601	2000	2001
contract	17	1001	1771	3434	2737	3434	3434
facturi	10	994	1741	3444	27380	34350	34350
detalii_factura	23	5927	12607	48262	191660	240450	480923

Ne propunem să vedem care este cel mai bun instrument de procesare a datelor, R sau PostgreSQL. Astfel, pentru a putea pune în evidență performanța celor două tehnologii de procesare a datelor, dimensiunea bazei de date a fost modificată treptat de la 9MB la 175MB. Au fost create o serie de subscheme ce au fost populate cu un anumit număr de înregistrări (Tabelul 2).

SQL: **R:**

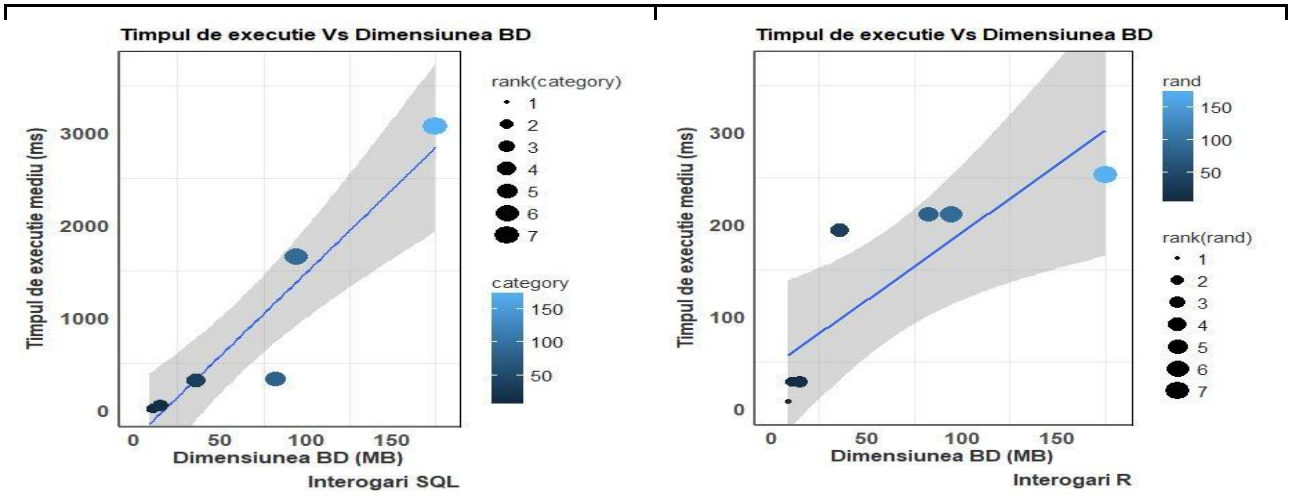


Figura 3. Timpul de execuție mediu în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL.

Performanța celor două sisteme a fost testată prin măsurarea și înregistrarea timpilor de execuție a interogărilor lansate în execuție. Au fost create mai multe tipuri de interogări (Tabelul 3) în care numărul de atribute din clauza SELECT și WHERE variază, respectiv numărul de

joncțiuni din clauza FROM variază în funcție de soluția găsită. De asemenea, în clauza WHERE utilizată în filtrarea înregistrărilor au fost utilizați și operatori logici precum AND și OR dar și operatori precum BETWEEN, IN sau operatori aritmetici.

În continuare prezentam rezultatele obținute în ceea ce privește performanța celor două tehnologii, PostgreSQL și R, utilizate pentru procesarea datelor colectate de o companie, prin lansarea în execuție a unui set de 100 de interogări (Tabelul 3) pe fiecare schemă în parte și înregistrarea timpilor de execuție. De asemenea, rezultatul returnat de fiecare interogare conține un număr de înregistrări ce variază în funcție de dimensiunea bazei de date.

Tabelul 3. Numărul de interogări testate din fiecare tip de interogare.

Tip interogare	Notatie interogare	Numar interogari
Selecție, proiecție, joncțiune	SPJ	34
Diferență, intersecție, reuniune	DIRPE	15
Grupări simple	GRUPARI_SIMPLE	18
Grupări	GRUPARI	14
Subconsultări în clauza FROM	SCCF	2
Subconsultări în clauza HAVING	SCCH	5
Subconsultări în clauza SELECT	SCCS	1
Subconsultări în clauza WHERE	SCCW	11
Funcții fereastră	WF	2
Expresii-tabelă	ET	7

Performanța celor două tehnologii este analizată din punct de vedere al procesării datelor. Pentru a putea compara rezultatele obținute s-a folosit funcția microbenchmark() din pachetul microbenchmark.

După cum se poate observa din Figura 3, timpul de execuție crește odată cu creșterea dimensiunii bazei de date, atât în cazul interogărilor lansate în execuție în R cât și în PostgreSQL. Constatăm, de asemenea, că performanța, în ceea ce privește procesarea datelor, este mai bună în cazul R-ului comparativ cu cea a PostgreSQL pentru care s-a obținut un timp de răspuns mediu mai mare.

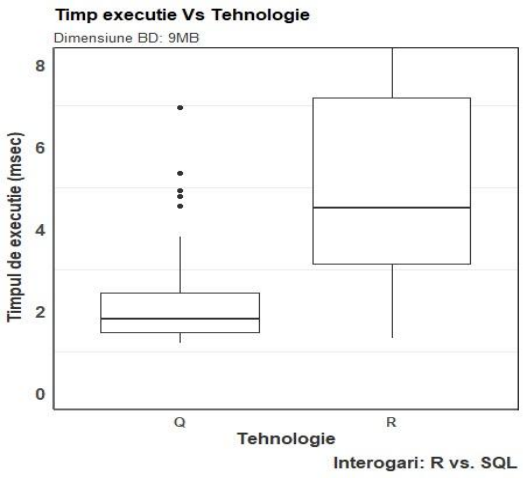
Tabelul 4. Diferențele privind performanța celor două tehnologii.

Dimensiunea bazei (MB)	$M_{ePostgreSQL}$	M_{eR}	$M_{ePostgreSQL}$-M_{eR}	$M_{ePostgreSQL}/M_{eR}$
9	1,81	4,51	-2,7	0,40
11	3,80	7,21	-4,41	0,46
15	5,90	8,30	-2,4	0,71
36	10,44	28,76	-18,32	0,36
82	23, 31	28,70	-5,39	0,81
94	43,32	28,72	14,62	1,51
175	83,50	61,48	22,02	1,36
Per total	**6,85**	**8,46**	**-1,61**	**0,81**

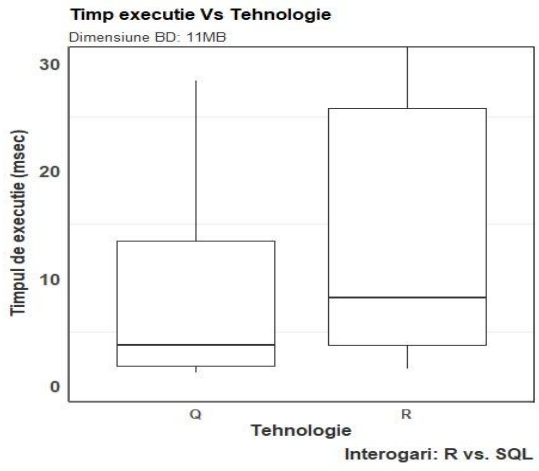
Figura 4 pune în evidență diferențele care apar între cele două tehnologii studiate odată cu creșterea dimensiunii bazei de date.

Astfel, diferența obținută între valorile timpilor de execuție (valorile medianei) în cazul celor două tehnologii studiate, PostgreSQL și R, variază pentru bazele de date cu dimensiunea de până la 82MB, pentru ca apoi să crească pentru bazele de date cu dimensiunea mai mare de 82MB și să devină mai performant R-ul.

SQL:



R:



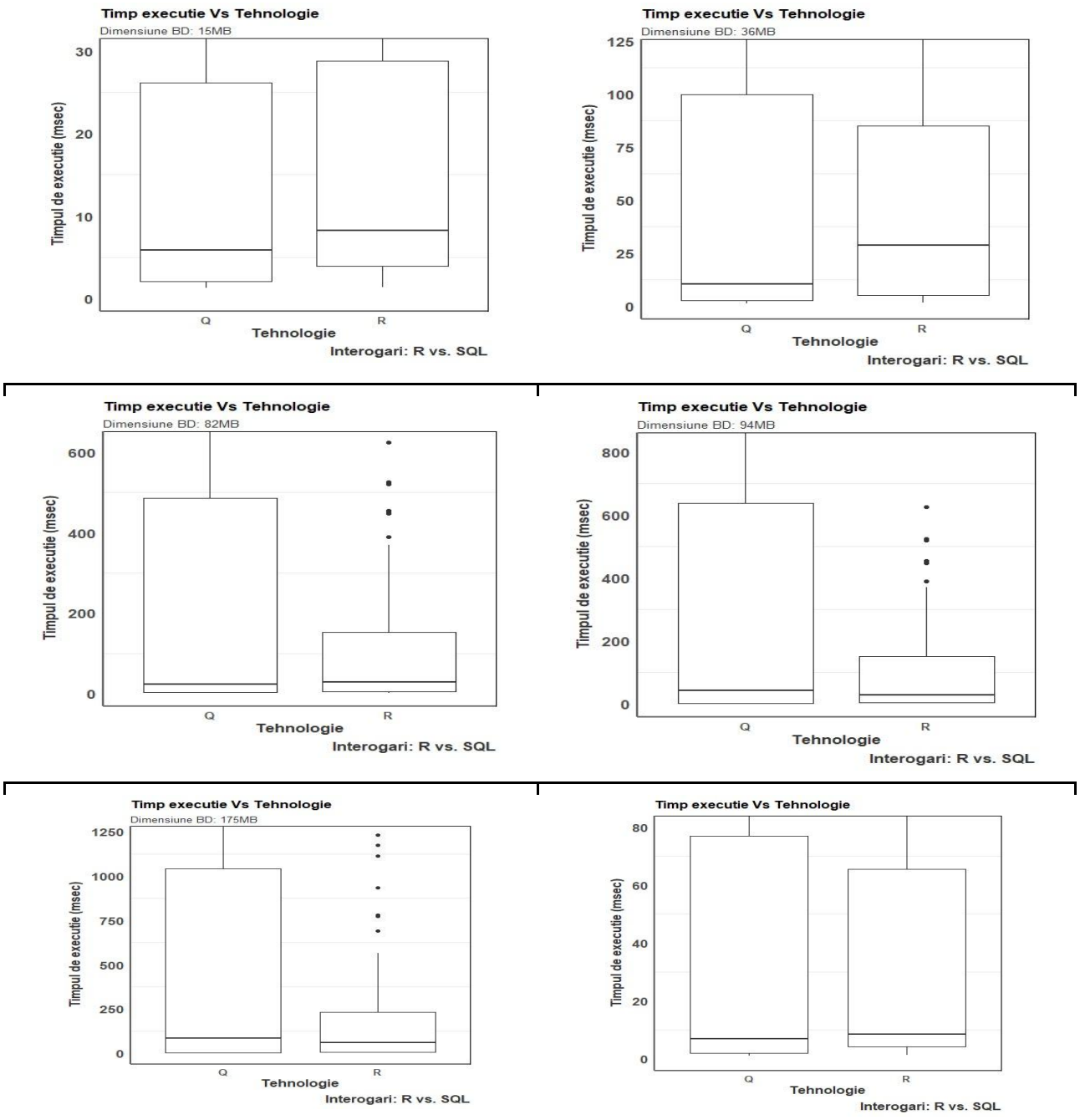


Figura 4. Valorile maxime, minime, media și mediana înregistrate pentru timpul de execuție în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL.

În Tabelul 4 sunt prezentate valorile medianei, diferența dintre valorile medianei și raportul dintre valorile medianei obținute pentru timpul de execuție a interogărilor testate în PostgreSQL și R, per total și pentru fiecare bază de date în parte.

Performanța PostgreSQL este mai bună față de cea a R-ului pentru bazele de date cu dimensiunea cuprinsă între 9 și 82MB, pentru ca situația să se schimbe, în cazul bazelor de date cu dimensiunea mai mare de 82MB, și să devină mai performant R-ul.

Un comportament similar a fost constatat și de I. Hrubaru și M. Fotache în lucrarea On the Performance of Three In-Memory Data Systems for On Line Analytical Processing în cadrul căreia sunt analizate trei platforme Oracle, SQL Server și MemSQL și care pune în evidență faptul că timpul de execuție a interogărilor rulate pe SQL Server configurația on-disk este mai mare comparativ cu cel al interogărilor rulate pe configurația in-memory.

În continuare vom face o analiza mai detaliată a timpului de execuție pentru fiecare tip de interogare în parte.

4. 1. Interogări selecție, proiecție și joncțiune

Rezultatele înregistrate pentru interogările de tip SPJ arată că odată cu creșterea dimensiunii bazei de date crește și valoarea timpului de execuție a interogărilor, iar diferențele înregistrate pentru valorile timpilor medii de execuție măsurați pentru bazele de date analizate, de asemenea, cresc.

Figura 5 pune în evidență, de asemenea, faptul că există diferențe și în ceea ce privește valoarea timpului de execuție înregistrat pentru interogările lansate în execuție în cadrul celor două tehnologii, R respectiv PostgreSQL. În acest caz, rezultatele obținute arată că R-ul este mai performant comparativ cu PostgreSQL, deoarece valorile înregistrate pentru timpul de execuție sunt de 10 ori mai mici.

SQL: **R:**

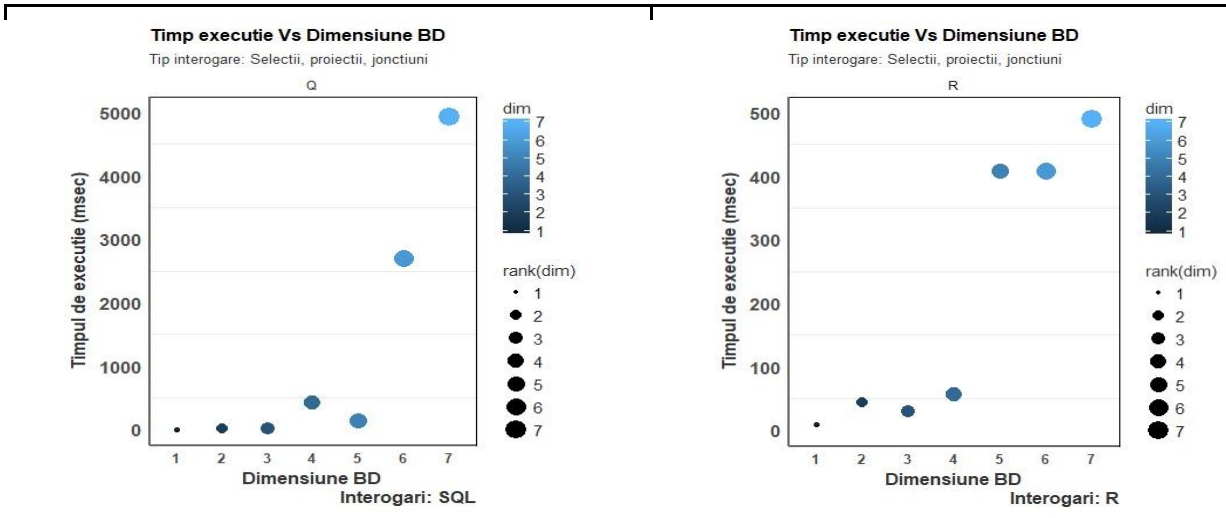


Figura 5. Timpul de execuție mediu în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL pentru interogările de tip SPJ.

La nivelul tuturor bazelor de date domeniul de valori înregistrat pentru timpul de execuție în cazul interogărilor SPJ lansate în execuție în PostgreSQL variază între 2,29 msec. și 4936,99 msec. cu o valoare medie a timpului de execuție de 1174,78 msec. și o valoare a medianei de 146, 12 msec. Pentru interogările lansate în execuție în R domeniul de valori obținut variază între 8,99 msec. și 489,50 msec. cu o valoare medie a timpului de execuție de 206,32 msec. și o valoare a medianei de 56,21 msec.

4. 2. Grupări

În cazul grupărilor după un singur atribut respectiv după două sau mai multe atribute trendul se păstrează. Astfel, domeniul de valori înregistrat pentru timpul de execuție a interogărilor cu grupări simple lansate în execuție în R variază între 5,63 msec. și 305,40 msec. cu o valoare medie a timpului de execuție de 120,65 msec. și o valoare a medianei de 60,70 msec., în timp ce pentru interogările cu grupări după două sau mai multe atribute domeniul de valori variază între 4,24 msec. și 380,31 msec. cu o valoare medie a tipului de execuție de 107,84 msec. și o valoare a medianei de 58,39 msec.

SQL: **R:**

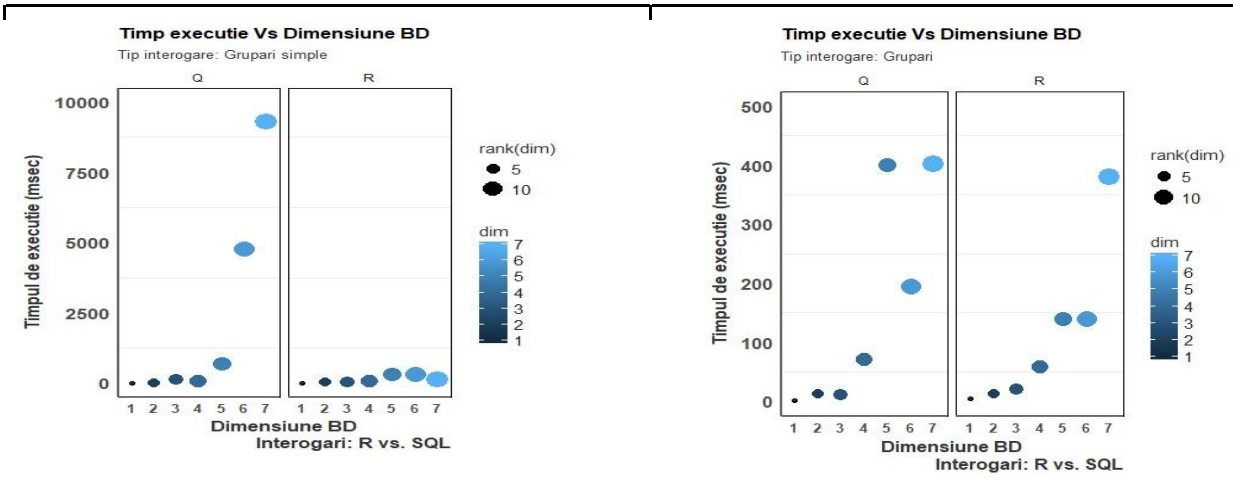


Figura 6. Timpul de execuție mediu în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL pentru interogările de tip Grupări_simple și Grupări.

Pentru interogările cu grupări simple lansate în execuție în PostgreSQL domeniul de valori obținut variază între 2,08 msec. și 9306,58 msec. pentru grupările simple cu o valoare medie a timpului de execuție de 2132,73 msec. și o valoare a medianei de 130,03 msec., în timp ce pentru interogările ce conțin grupări după două sau mai multe variabile domeniul de valori

variază între 1,86 msec. și 401,99 cu o valoare medie de 155,88 msec. și o valoare a medianei de 69,89 msec (Figura 6).

4. 3. Diferențe, reuniuni și intersecții

SQL: **R:**

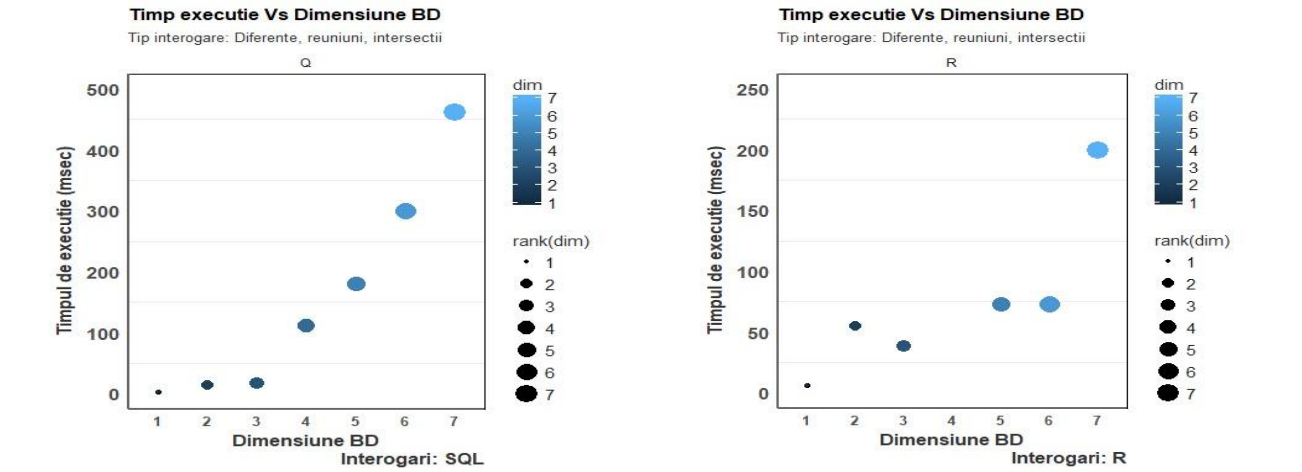


Figura 7. Timpul de execuție mediu în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL pentru interogările de tip DIRPE

În cazul interogărilor de tip reuniune, intersecție și diferență observăm că diferențele în ceea ce privește timpul de execuție a interogărilor nu sunt foarte mari, însă și în această situație performanța R-ul în ceea ce privește procesarea datelor este mai bună (Figura 7 si Figura 8) pentru bazele de date cu dimensiunea mai mare de 36MB (Tabelul 5).

SQL: **R:**

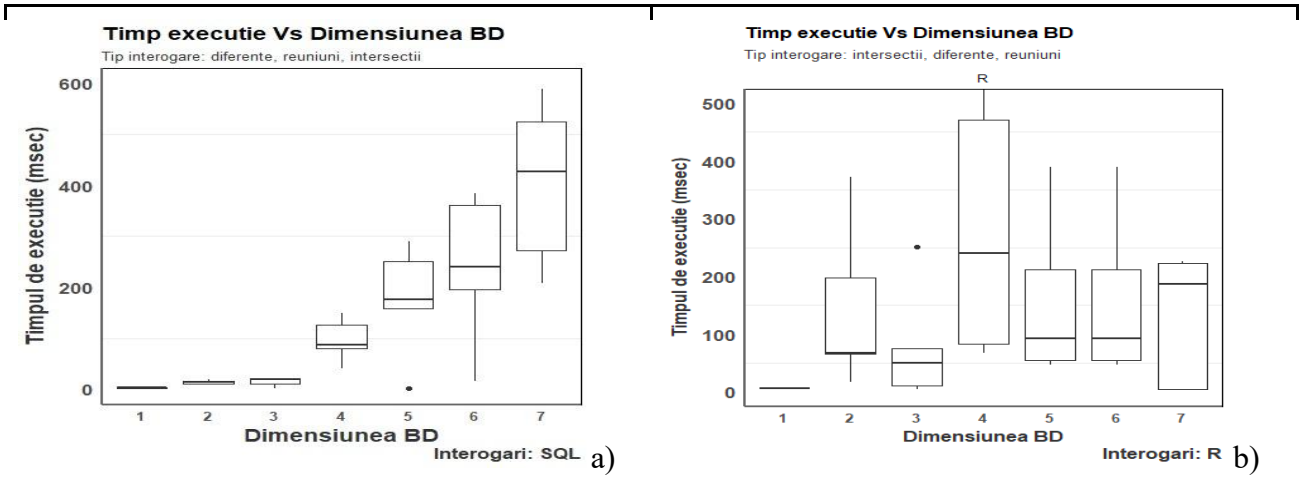


Figura 8. Timpii de execuție minim, maxim, mediu și mediana în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL pentru interogările de tip DERPI: a) pentru interogările rulate în R și b) pentru interogările rulate în SQL.

Tabelul 5. Valorile obținute pentru timpul de execuție minim, maxim, mediu, valoarea medianei și $M_{ePostgreSQL}/M_{eR}$.

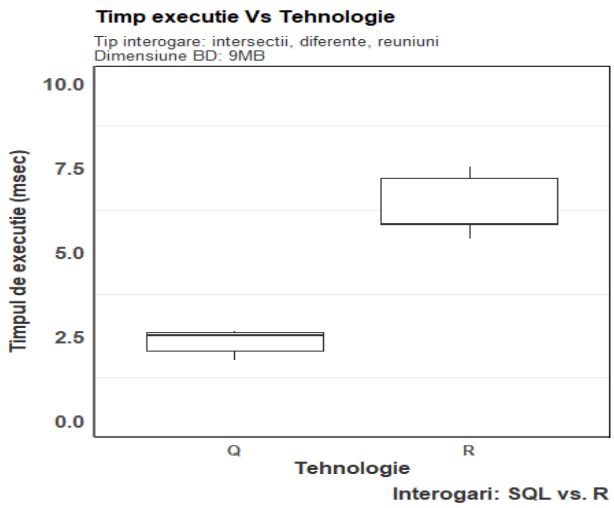
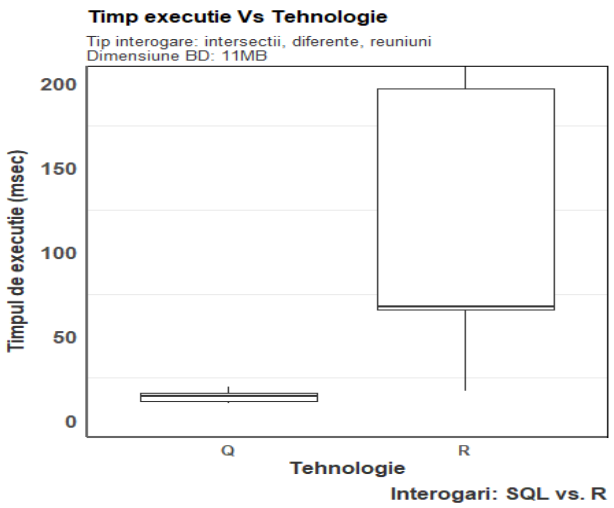
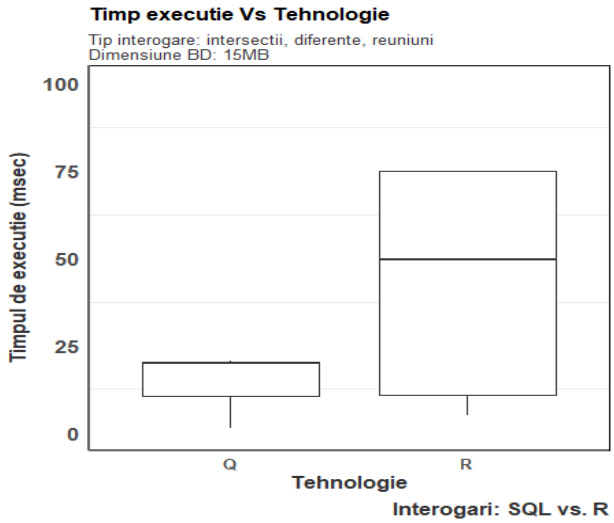
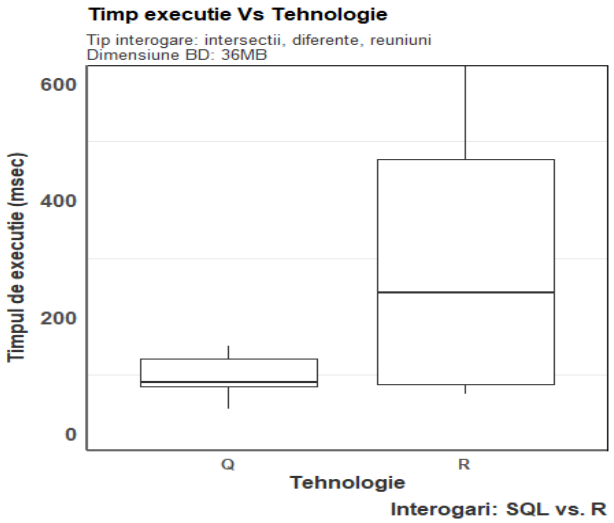
Tip interogare	Dimensiune BD (MB)	Tehnologie	Timpul min. (msec.)	Timpul max. (msec.)	Timpul mediu (msec.)	Mediana (msec.)	$M_{ePostgreSQL}$-M_{eR}	$M_{ePostgreSQL}/M_{eR}$
DIRPE	9	SQL	1,8	2,6	2,3	2,5	-3,3	0,43
		R	4,2	7,5	5,9	5,8		
	11	SQL	10,0	20,0	14,3	14,2	-52,4	0,21
		R	2,0	371,9	120,2	66,6		
	15	SQL	1,3	20,5	14,5	19,9	-12,8	0,61
		R	4,9	68,6	67,9	32,7		
	36	SQL	40,1	149,8	96,5	86,7	-268,8	0,24
		R	66,6	651,7	350,8	355,5		
	82	SQL	1,7	290,3	175,6	175,9	93,8	2,14
		R	46,41	389,2	144,3	82,1		
	94	SQL	16,4	384,1	239,6	241,1	159,0	2,94
		R	46,4	389,2	144,3	82,1		
	175	SQL	209,0	588,5	404,5	426,7	226,4	2,13
		R	3,2	226,1	142,9	200,3		
	Per total	**SQL**	**1,3**	**588,5**	**135,3**	**40,1**	**29,7**	**0,57**
		R	**1,9**	**651,7**	**139,5**	**69,8**		

În ceea ce privește bazele de date cu dimensiunea mai mică de 36MB observăm că performanța PostgreSQL-ului este mai bună.

Astfel, per total, pentru interogările de tip DIRPE lansate în execuție în R, domeniul de valori a timpului de răspuns per total variază între 1,9 msec. și 651,7 msec. cu o valoare a mediei de 139,5 msec. și o valoare a medianei de 69,8 msec.

În ceea ce privește interogările rulate în PostgreSQL timpul de răspuns variază între 1,3 msec. și 588,5 msec. cu o valoare medie de 135,3 msec. și o valoare a medianei de 40,1 msec (Figura 9) ceea ce arată că performanța PostgreSQL este mai bună.

În Tabelul 5 sunt prezentate valorile timului de execuție/răspuns minim, maxim, mediu, mediana, diferența dintre valorile mediane obținute pentru cele doua tehnologii, și raportul valorilor medianei obținut pentru cele două tehnologii în funcție de dimensiunea bazei de date.

SQL:	**R:**
	
	

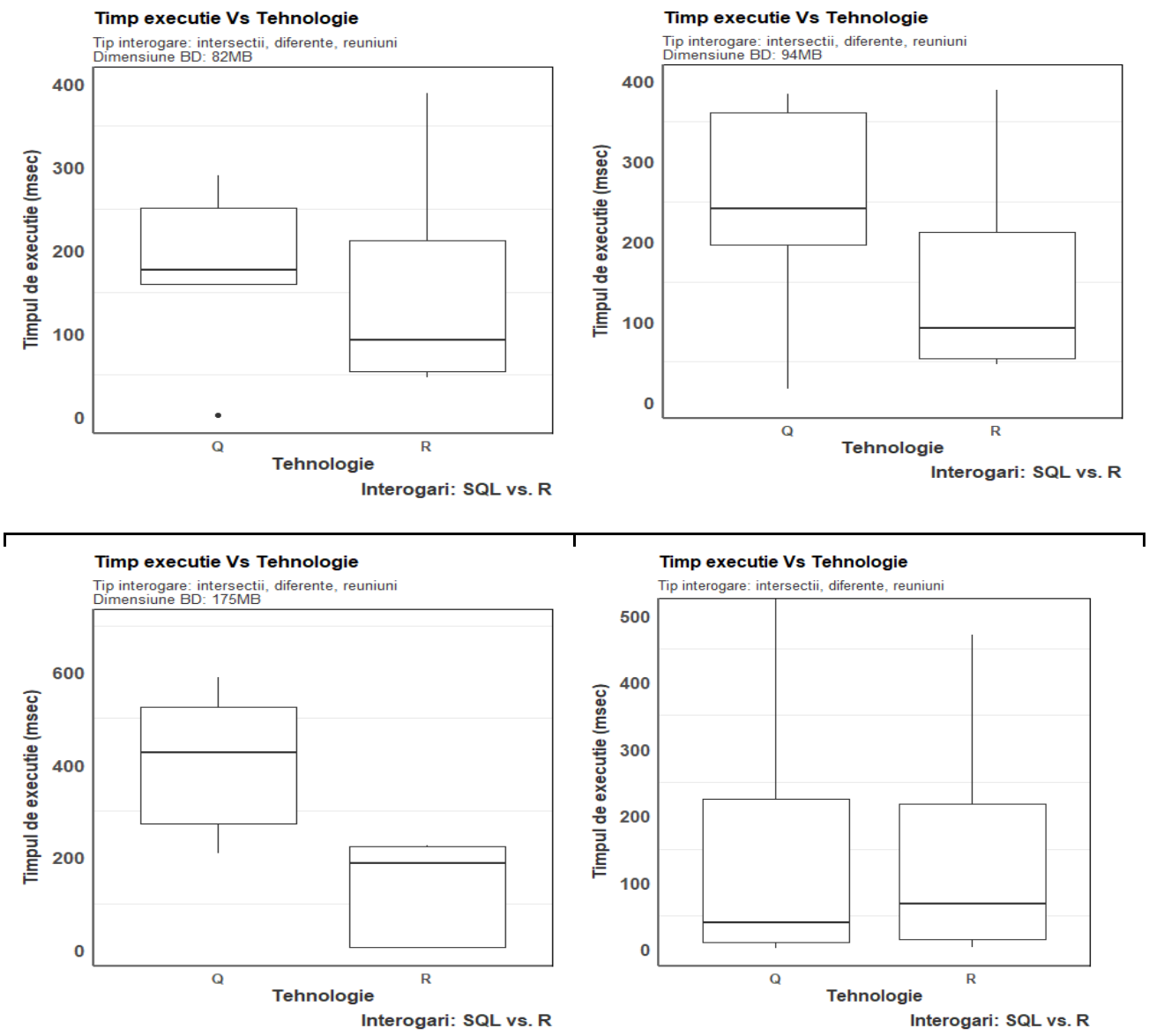


Figura 9. Comparații între valorile obținute pentru timpii de execuție minim, maxim, mediu și mediana măsurați în R și PostgreSQL în funcție de dimensiunea bazei de date pentru interogările de tip DERPI.

4. 4. Subinterogări în clauza SELECT, FROM, WHERE, HAVING

În ceea ce privește interogările ce conțin subconsultări în clauza SELECT, FROM, WHERE și HAVING rezultatele obținute sunt prezentate în Figura 10. În toate cazurile se poate observa că tendința se menține, R-ul fiind mai performant în ceea ce privește procesarea datelor.

În Tabelul 6 sunt prezentate valorile obținute pentru timpul de execuție minim, maxim, timpul de execuție mediu și valoarea medianei în R și PostgreSQL.

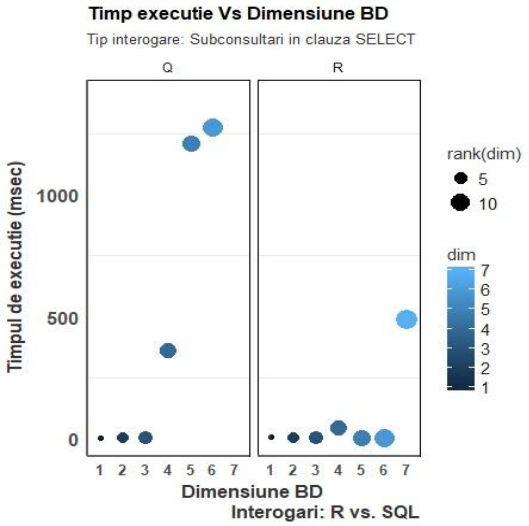
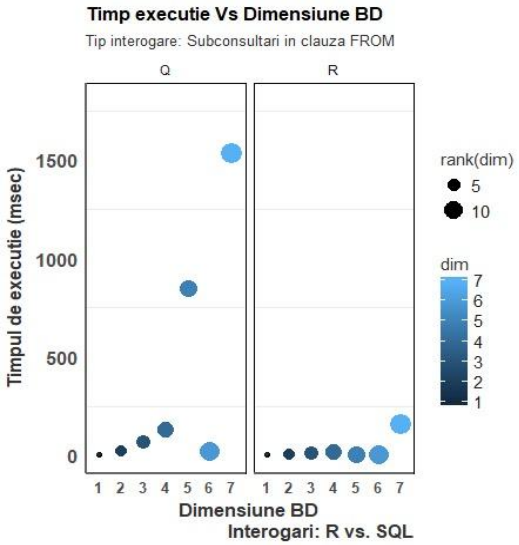
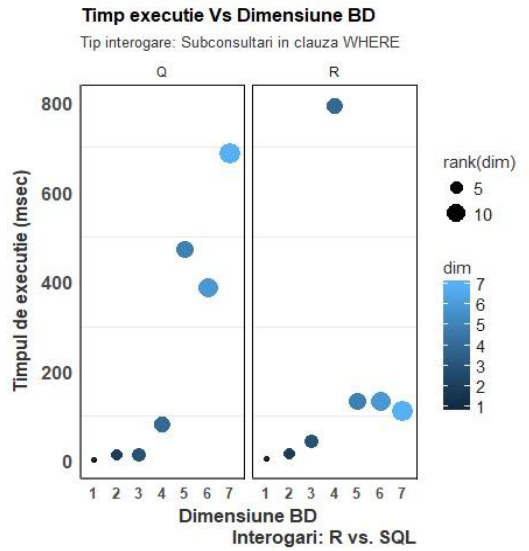
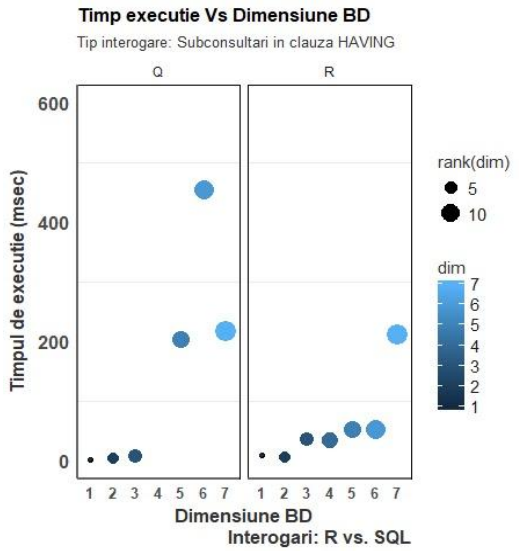
SQL:	R:
	
	

Figura 10. Timpul de execuție în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL pentru interogările ce conțin subconsultări în clauza SELECT, FROM, WHERE și HAVING.

Astfel, pentru interogările de tip SCCS, SCCF, SCCW și SCCH observăm că pentru bazele de date cu dimensiunea de 9MB, 11MB și 15MB performanța PostgreSQL este mai bună, în timp ce pentru baze de date cu dimensiunea mai mare de 15MB performanța R-ului devine vizibil mai bună (Figura 10).

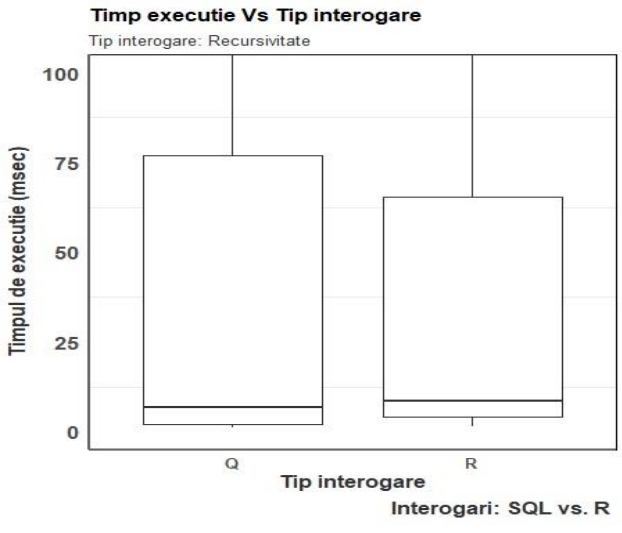
Tabelul 6. Valorile obținute pentru timpul de execuție minim, maxim mediu și valoarea medianei.

Tip interogare	Tehnologie	Timpul min. (msec.)	Timpul max. (msec.)	Timpul mediu (msec.)	Mediana (msec.)
SELECT	R	1,8	486,3	78,6	5,6
	SQL	1,7	2420,1	753,6	360,4
FROM	R	3.5	158,9	29,7	6,5
	SQL	1,9	1532,5	375,2	69,5
WHERE	R	5,3	791,0	175,5	110,3
	SQL	2,1	686,6	80,3	235,8
HAVING	R	5,7	210,2	57,1	35,8
	SQL	1,96	3127,9	573,8	204,1

4. 5. Recursivitate

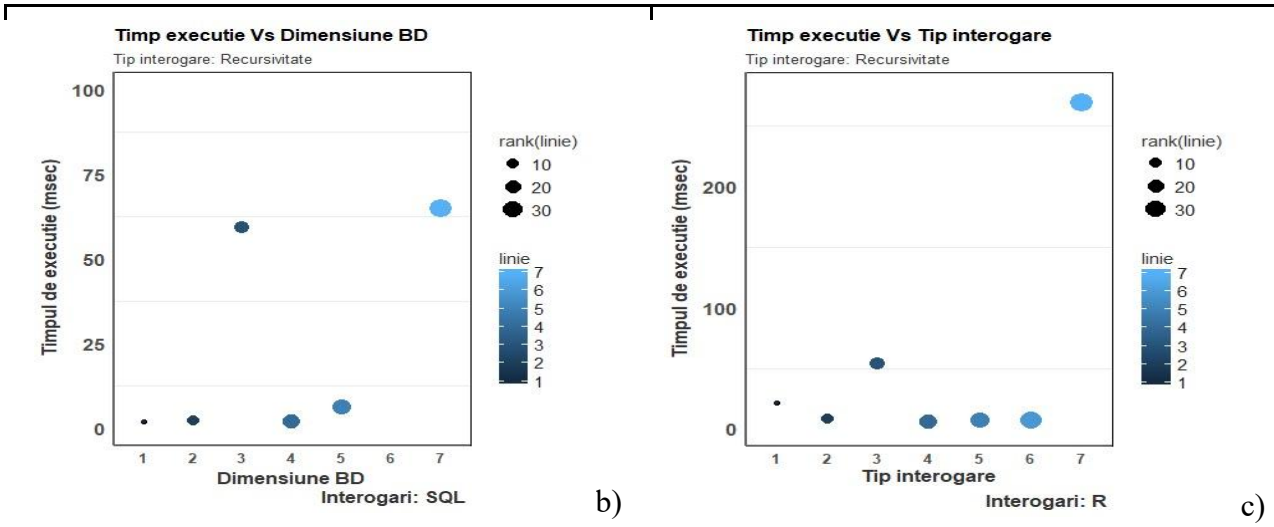
În ceea ce privește interogările recursive observăm că în R domeniul de valori variază între 7, 11 msec. și 269,47 msec., cu o valoare medie a timpului de execuție de 53,22 msec. și o valoarea a medianei de 8, 21 msec., în timp ce în PostgreSQL domeniul de valori înregistrat

SQL vs. R



a)

SQL: **R:**



b) c)

Figura 11. Timpul de execuție în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL pentru interogările recursive

pentru timpul de execuție variază între 2,02 msec. și 352,22 msec. cu o valoare medie a timpului de execuție de 69,86 msec. și o valoare a medianei de 6,26 msec (Figura 11 a)).

Pentru interogările rulate pe bazele de date cu dimensiunea de 9MB, 11MB, 36MB și 175MB timpul de răspuns este mai mic comparativ cu interogările rulate pe bazele de date cu dimensiunea de 15MB și 94MB pentru care s-au înregistrat valori mai mari pentru timpii de răspuns (Figura 11 b) și c)).

Per total, pentru acest tip de interogare, se poate observa că performanta PostgreSQL este mai bună comparativ cu cea a R.

4. 6. Funcții fereastră și expresii-tabelă

În Figura 12 sunt prezentate rezultatele obținute pentru interogările de tip expresii-tabelă și OLAP. Și în acest caz tendința se păstrează, în sensul că timpul de execuție a interogărilor crește cu creșterea dimensiunii bazei de date. La interogările de tip OLAP observăm că pentru baza de date cu dimensiunea de 175MB timpul de răspuns pentru interogările rulate în R respectiv SQL scade foarte mult.

SQL: **R:**

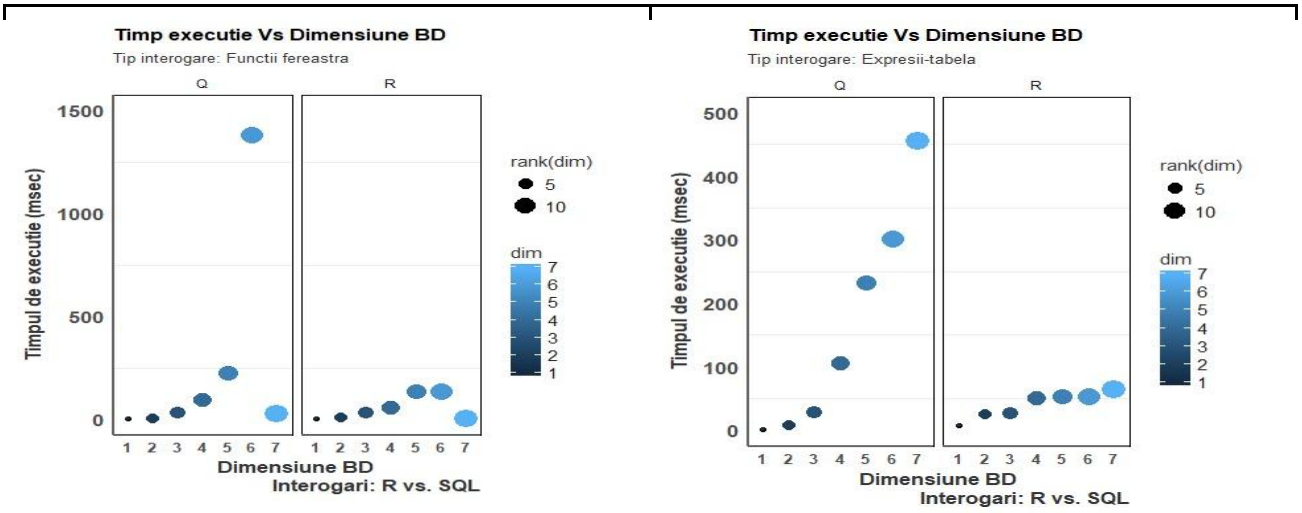


Figura 12. Timpul de execuție în funcție de dimensiunea bazei de date măsurat în R și PostgreSQL pentru interogările de tip WF și ET.

Obținem, astfel, pentru interogările rulate în R un domeniu de valori al timpului de execuție cuprins între 2 msec. și 5,6 msec. cu o valoare pentru timpul mediu de 4,2 msec. și o valoare a medianei de 5,6 msec. În ceea ce privește interogările rulate în PostgreSQL acestea au un domeniu de valori cuprins între 1,7 msec. și 63,0 msec. cu o valoarea pentru timpul de execuție mediu de 29,6 msec. și o valoare a medianei de 1,76 msec.

Pentru îmbunatățirea timpului de răspuns/execuție a interogărilor pot fi utilizate cheile primare, indecși, diferite tehnici de optimizare precum utilizarea în clauza WHERE, ORDER BY și JOIN a unor indecși creați pe baza predicatelor[85] utilizate etc.

Tehnicile de optimizare a interogărilor sunt extrem de importante atât pentru administratori de baze de date cât și pentru furnizori de SGBD-uri. Pentru a îmbunătăți performanța interogărilor formulate în SQL și R, administratorii de baze de date trebuie să înțeleagă instrumentele disponibile pe platformele SQL și R ce furnizează informații cu privire la planul de execuție al interogării. Cea mai bună metodă de optimizare a unei interogări se realizează prin scrierea acelei interogări în mai multe moduri și compararea timpului de execuție și a planului de execuție a soluțiilor propuse.

[85] Hrubaru, I., Fotache, M., On the Performance of Three In-Memory Data Systems for On Line Analytical Processing, Informatica Economica vol. 21, no. 1, 2017, p. 13.

Înregistrarea timpului de execuție a interogărilor inițiale și a celor optimizate permite compararea valorilor obținute pentru timpi de execuție evidențiind astfel timpului de răspuns îmbunătățit a interogările optimizate.

J. Habimana în lucrarea Query Optimization Techniques - Tips For Writing Efficient And Faster SQL Queries oferă o serie de sugestii cu privire la scrierea unei interogări eficiente (tehnici de optimizare a interogărilor), precum[86]: scrierea listei de atribute în clauza SELECT pentru care dorim să vizualizăm valorile în locul scrierii SELECT * deoarece dimensiunea tabelului rezultat este mai mică; evitarea utilizării clauzei HAVING în fraza SELECT; utilizarea clauzei DISTINCT, eliminarea subinterogărilor din clauza WHERE, folosirea operatorului IN în loc de OR atunci când folosim coloane indexate în clauza WHERE, utilizarea clauzei UNION ALL în loc de UNION deoarece prima variantă este mai rapidă decât a doua, evitarea utilizării operatorului logic OR în JOIN, evitarea utilizării funcțiilor sau metodelor de partea dreapta a operatorilor folosiți în clauza WHERE, evitarea coloanelor calculate în clauza WHERE etc.

Optimizarea interogărilor reprezintă o sarcină importantă atât pentru administratori de baze de date cât și pentru furnizori de SGBD-uri întrucât permite îmbunătățirea performanței SGBD-ului. Prin urmare, modul în care sunt scrise interogările poate influența performanța SGBD-ului și implicit a timpului de răspuns/execuție a interogărilor.

[86] Habimana, J., Query Optimization Techniques - Tips For Writing Efficient And Faster SQL Queries, International journal of scientific & technology research, 4, 10, 2015.

Concluzii

Atât SGBD-urile, proprietare (cum ar fi: Oracle, Microsoft SQL etc.) sau open-source (PostgreSQL, MySQL etc.) cât și R-ul pot fi privite ca instrumente ce permit interogarea și procesarea volumelor de date colectate. Limbajul SQL permite extragerea datelor prin formularea de interogări SQL. În timp ce, pentru realizarea unor analize mai complexe (analize descriptive, analize predictive, reprezentări grafice etc.) este necesară utilizarea unor instrumente de analiză specializate, precum R.

Deși la prima vedere cele două limbaje par a fi complet diferite și utilizate în scopuri diferite acestea au câteva caracteristici comune. Limbajul SQL este utilizat pentru interogarea și actualizarea datelor stocate în bazele de date relaționale, în timp ce limbajul R este utilizat pentru accesarea și analiza datelor. Inițial limbajul SQL nu a fost proiectat pentru analiza datelor, ci mai degrabă pentru optimizarea, interogarea și actualizarea datelor. Analiza datelor cu ajutorul limbajului SQL necesită construirea de interogări complicate, iar numărul joncțiunilor între tabele poate influența timpul de răspuns/vitezei de răspuns a interogărilor lansate. Spre deosebire de SQL, limbajul R este dedicat analizei și procesării datelor.

R-ul nu poate fi considerat o alternativă a sistemelor de gestiune a bazelor de date deoarece acesta încarcă toate datele ce urmează a fi procesate în memoria RAM, și nu stochează datele sau rezultatele prelucrării. De asemenea, R-ul nu permite optimizarea interogărilor ceea ce poate duce la scăderea vitezei de răspuns/creșterea timpului de răspuns.

În cadrul lucrării s-a analizat/testat performanța celor două sisteme prin măsurarea și înregistrarea timpilor de execuție a interogărilor lansate în execuție. Au fost testate un număr de 100 de interogări care au fost impărțite pe tipuri de interogări. În urma analizei s-a constatat că timpul de execuție este direct proporțional cu volumul datelor prelucrate. De asemenea, apar diferențe și în ceea ce privește tehnologia de analiză utilizată. Astfel, pentru baze de date mai mici de 82MB, PostgreSQL este mai performant în ceea ce privește procesarea datelor. În timp ce pentru baze de date ce depășesc dimensiunea de 82MB devine mai performant R-ul.

Cele două limbaje de procesare, R și SQL, sunt utilizate în cazuri/situații diferite. Limbajul SQL este utilizat preponderent în stocarea datelor critice ale organizațiilor, permițând utilizatorilor accesarea, modificare, inserarea și ștergerea datelor într-un mediu centralizat, în timp ce limbajul R este utilizat pentru lucrul cu datele (procesarea, analizarea și vizualizarea). O parte din operații sunt mai ușor de realizat în SQL altele nu deoarece crește complexitatea interogărilor.

Anexa 1

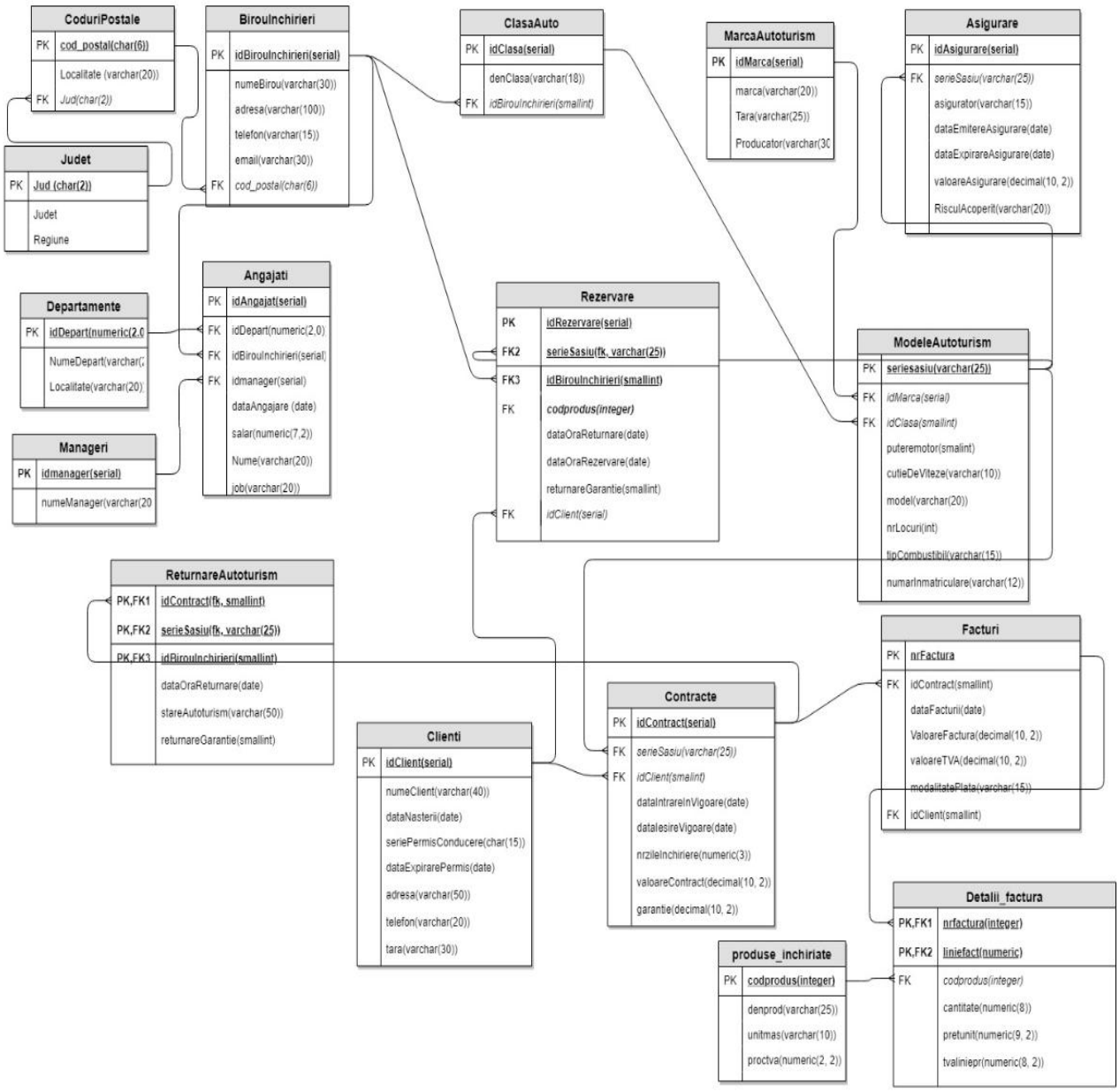


Figura 1. Structura/schema bazei de date “Inchirieri masini”.

Anexa 2

Interogări selecție, proiecție și joncțiune

SQL: **R:**

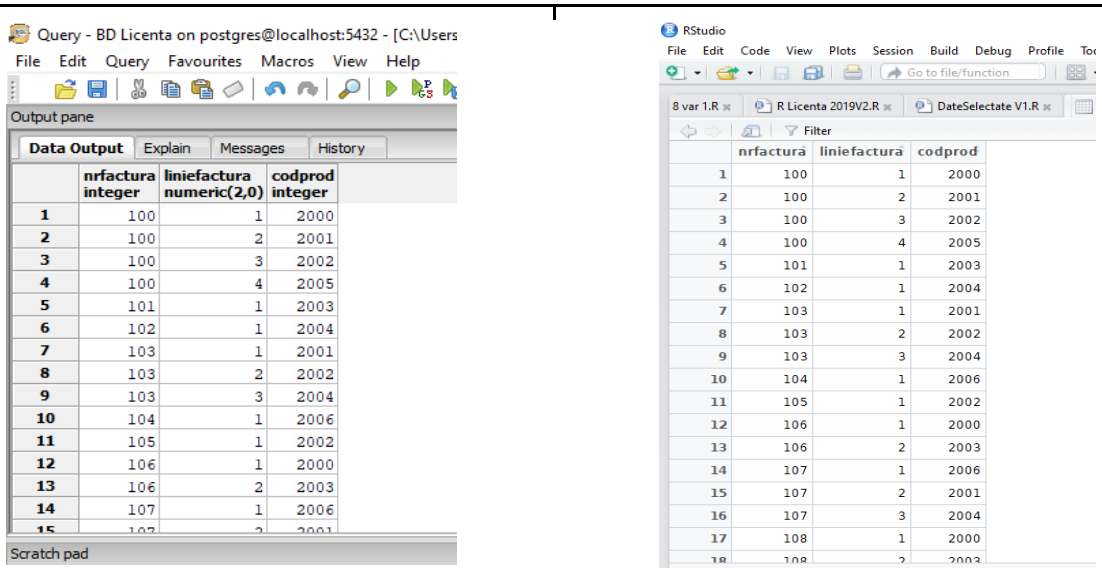
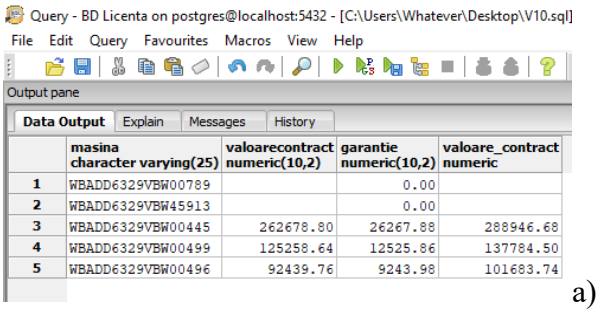


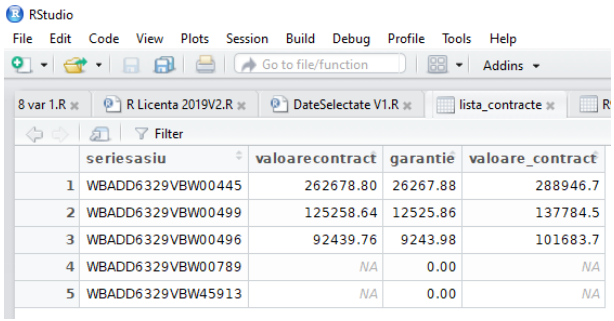
Figura 1. Afișarea informațiilor privind numărul facturi, poziția pe factură și codul produselor conținute de fiecare factură.

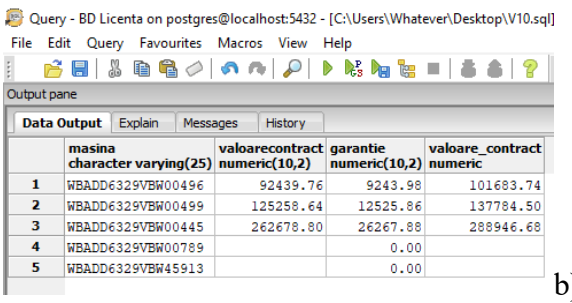
Ordonarea atributelor

SQL: **R:**



a)





b)

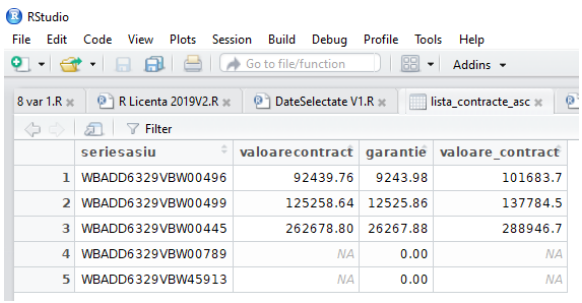
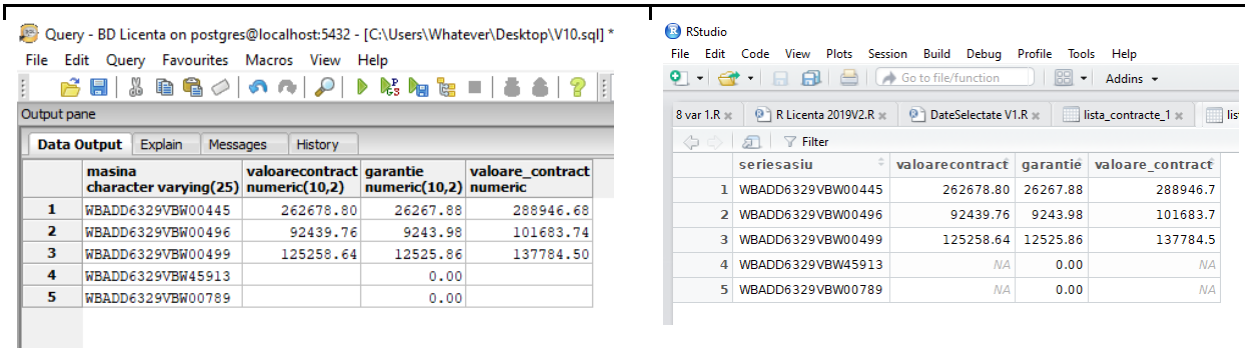


Figura 2. Afișarea informațiilor privind contractele încheiate cu diverși clienți în anul 2017: a) listă ordonată descrescător și b) listă ordonată crescător după atributul valoareacontract.

Ordonare după mai multe variabile

SQL: **R:**



Query - BD Licenta on postgres@localhost:5432 - [C:\Users\Whatever\Desktop\V10.sql] *

File Edit Query Favourites Macros View Help

Output pane

Data Output | Explain | Messages | History

	masina character varying(25)	valoarecontract numeric(10,2)	garantie numeric(10,2)	valoare_contract numeric
1	WBADD6329VBW00445	262678.80	26267.88	288946.68
2	WBADD6329VBW00496	92439.76	9243.98	101683.74
3	WBADD6329VBW00499	125258.64	12525.86	137784.50
4	WBADD6329VBW45913		0.00	
5	WBADD6329VBW00789		0.00	

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

8 var 1.R | R Licenta 2019V2.R | DateSelectate V1.R | lista_contracte_1 | lis

Filter

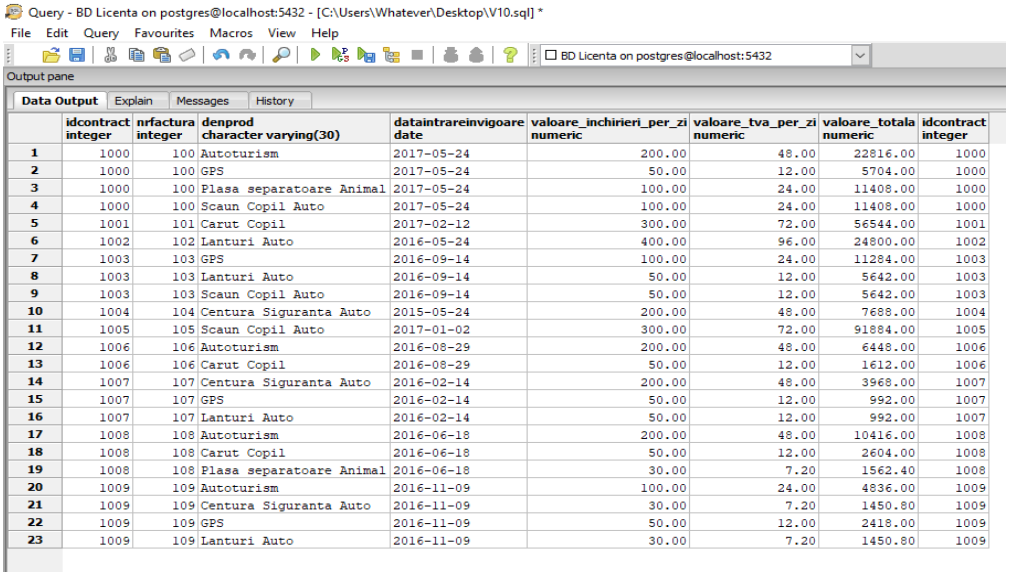
	seriesasiu	valoarecontract	garantie	valoare_contract
1	WBADD6329VBW00445	262678.80	26267.88	288946.7
2	WBADD6329VBW00496	92439.76	9243.98	101683.7
3	WBADD6329VBW00499	125258.64	12525.86	137784.5
4	WBADD6329VBW45913	NA	0.00	NA
5	WBADD6329VBW00789	NA	0.00	NA

Figura 3. Ordonarea înregistrărilor crescătoare după atributele dataiesirevigoare și nrzileinchiriere și descrescătoare după atributul "valoare_contract".

Joncțiuni. Tipuri de joncțiuni

Joncțiunea internă (inner join)

SQL:



Query - BD Licenta on postgres@localhost:5432 - [C:\Users\Whatever\Desktop\V10.sql] *

File Edit Query Favourites Macros View Help

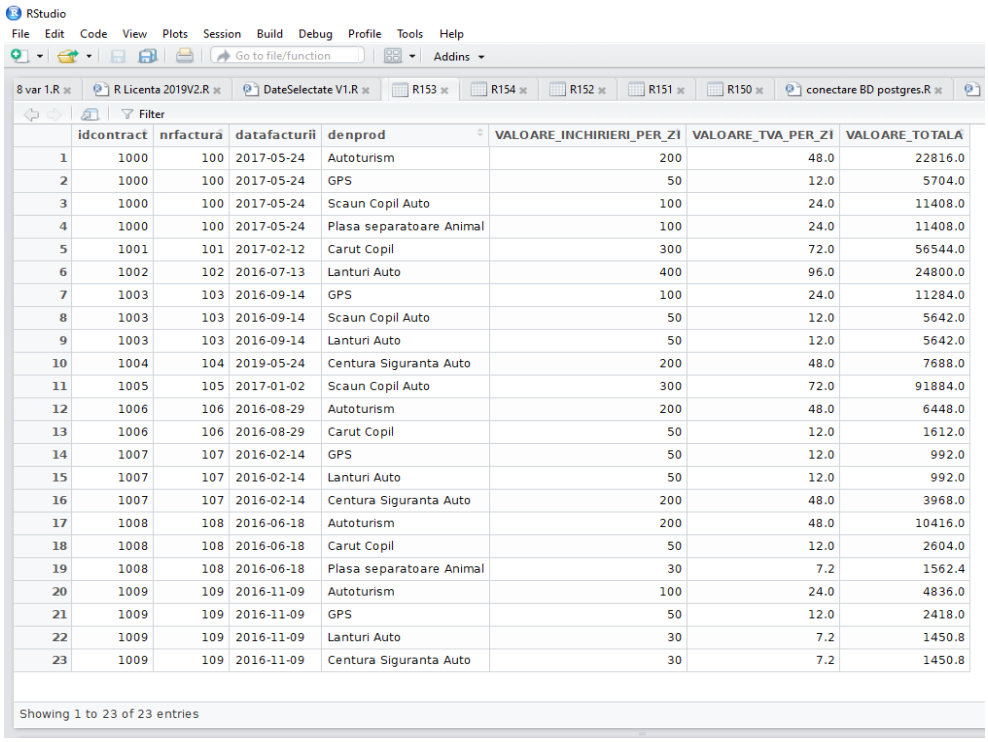
BD Licenta on postgres@localhost:5432

Output pane

Data Output | Explain | Messages | History

	idcontract integer	nrfactura integer	denprod character varying(30)	dataintrareinvigoare date	valoare_inchirieri_per_zi numeric	valoare_tva_per_zi numeric	valoare_totala numeric	idcontract integer
1	1000	100	Autoturism	2017-05-24	200.00	48.00	22816.00	1000
2	1000	100	GPS	2017-05-24	50.00	12.00	5704.00	1000
3	1000	100	Plasa separatoare Animal	2017-05-24	100.00	24.00	11408.00	1000
4	1000	100	Scaun Copil Auto	2017-05-24	100.00	24.00	11408.00	1000
5	1001	101	Carut Copil	2017-02-12	300.00	72.00	56544.00	1001
6	1002	102	Lanturi Auto	2016-05-24	400.00	96.00	24800.00	1002
7	1003	103	GPS	2016-09-14	100.00	24.00	11284.00	1003
8	1003	103	Lanturi Auto	2016-09-14	50.00	12.00	5642.00	1003
9	1003	103	Scaun Copil Auto	2016-09-14	50.00	12.00	5642.00	1003
10	1004	104	Centura Siguranta Auto	2015-05-24	200.00	48.00	7688.00	1004
11	1005	105	Scaun Copil Auto	2017-01-02	300.00	72.00	91884.00	1005
12	1006	106	Autoturism	2016-08-29	200.00	48.00	6448.00	1006
13	1006	106	Carut Copil	2016-08-29	50.00	12.00	1612.00	1006
14	1007	107	Centura Siguranta Auto	2016-02-14	200.00	48.00	3968.00	1007
15	1007	107	GPS	2016-02-14	50.00	12.00	992.00	1007
16	1007	107	Lanturi Auto	2016-02-14	50.00	12.00	992.00	1007
17	1008	108	Autoturism	2016-06-18	200.00	48.00	10416.00	1008
18	1008	108	Carut Copil	2016-06-18	50.00	12.00	2604.00	1008
19	1008	108	Plasa separatoare Animal	2016-06-18	30.00	7.20	1562.40	1008
20	1009	109	Autoturism	2016-11-09	100.00	24.00	4836.00	1009
21	1009	109	Centura Siguranta Auto	2016-11-09	30.00	7.20	1450.80	1009
22	1009	109	GPS	2016-11-09	50.00	12.00	2418.00	1009
23	1009	109	Lanturi Auto	2016-11-09	30.00	7.20	1450.80	1009

R:



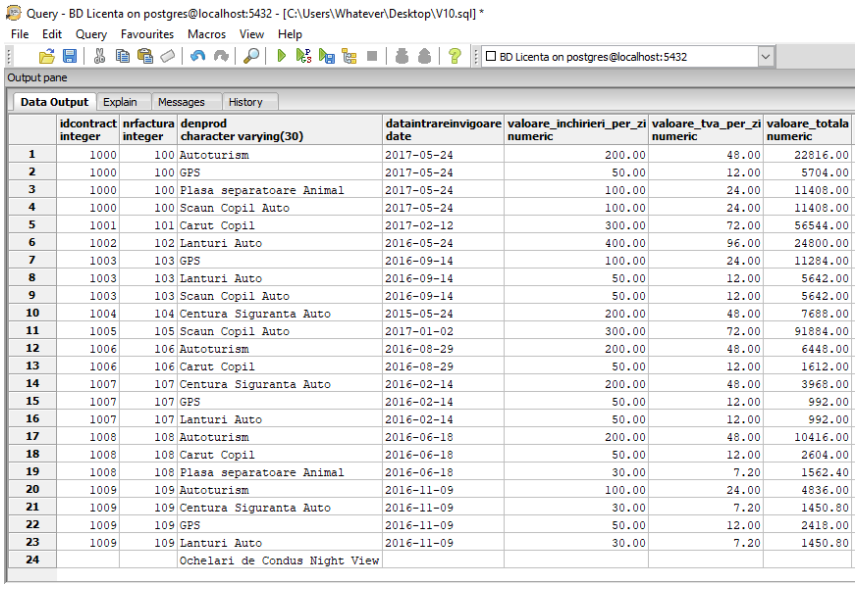
	idcontract	nrfactura	datafacturii	denprod	VALOARE_INCHIRIERI_PER_ZI	VALOARE_TVA_PER_ZI	VALOARE_TOTALA
1	1000	100	2017-05-24	Autoturism	200	48.0	22816.0
2	1000	100	2017-05-24	GPS	50	12.0	5704.0
3	1000	100	2017-05-24	Scaun Copil Auto	100	24.0	11408.0
4	1000	100	2017-05-24	Plasa separatoare Animal	100	24.0	11408.0
5	1001	101	2017-02-12	Carut Copil	300	72.0	56544.0
6	1002	102	2016-07-13	Lanturi Auto	400	96.0	24800.0
7	1003	103	2016-09-14	GPS	100	24.0	11284.0
8	1003	103	2016-09-14	Scaun Copil Auto	50	12.0	5642.0
9	1003	103	2016-09-14	Lanturi Auto	50	12.0	5642.0
10	1004	104	2019-05-24	Centura Siguranta Auto	200	48.0	7688.0
11	1005	105	2017-01-02	Scaun Copil Auto	300	72.0	91884.0
12	1006	106	2016-08-29	Autoturism	200	48.0	6448.0
13	1006	106	2016-08-29	Carut Copil	50	12.0	1612.0
14	1007	107	2016-02-14	GPS	50	12.0	992.0
15	1007	107	2016-02-14	Lanturi Auto	50	12.0	992.0
16	1007	107	2016-02-14	Centura Siguranta Auto	200	48.0	3968.0
17	1008	108	2016-06-18	Autoturism	200	48.0	10416.0
18	1008	108	2016-06-18	Carut Copil	50	12.0	2604.0
19	1008	108	2016-06-18	Plasa separatoare Animal	30	7.2	1562.4
20	1009	109	2016-11-09	Autoturism	100	24.0	4836.0
21	1009	109	2016-11-09	GPS	50	12.0	2418.0
22	1009	109	2016-11-09	Lanturi Auto	30	7.2	1450.8
23	1009	109	2016-11-09	Centura Siguranta Auto	30	7.2	1450.8

Showing 1 to 23 of 23 entries

Figura 4. Rezultatul joncțiunii internă INNER JOIN a patru table în SQL și R.

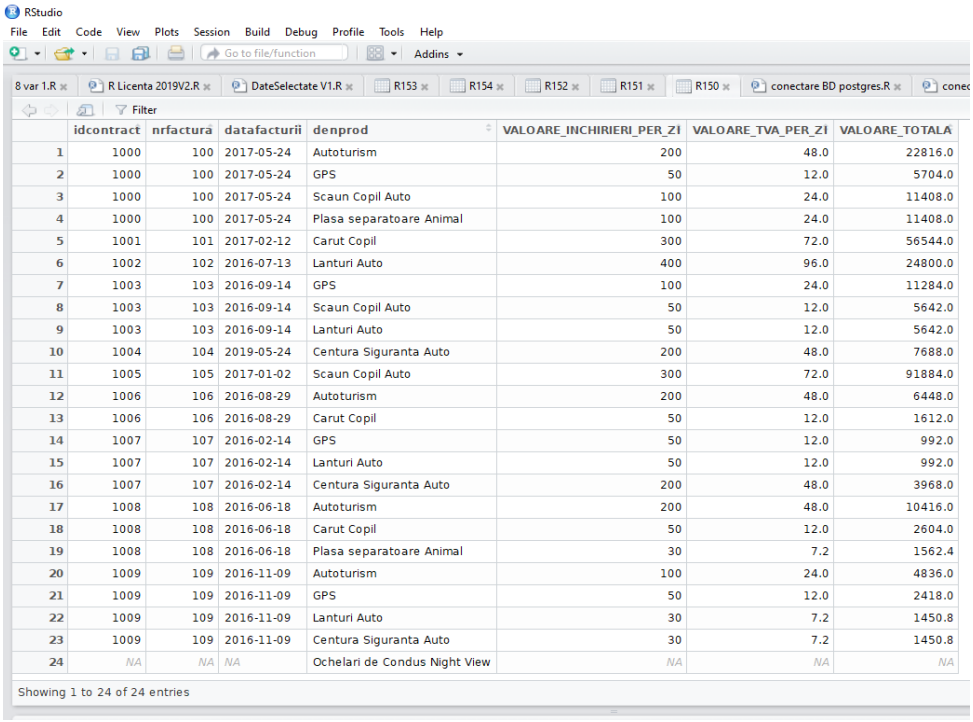
Joncțiunea externă stânga (LEFT OUTER JOIN)

SQL:



	idcontract integer	nrfactura integer	denprod character varying(30)	dataintrareinvigoare date	valoare_inchirieri_per_zi numeric	valoare_tva_per_zi numeric	valoare_totala numeric
1	1000	100	Autoturism	2017-05-24	200.00	48.00	22816.00
2	1000	100	GPS	2017-05-24	50.00	12.00	5704.00
3	1000	100	Plasa separatoare Animal	2017-05-24	100.00	24.00	11408.00
4	1000	100	Scaun Copil Auto	2017-05-24	100.00	24.00	11408.00
5	1001	101	Carut Copil	2017-02-12	300.00	72.00	56544.00
6	1002	102	Lanturi Auto	2016-05-24	400.00	96.00	24800.00
7	1003	103	GPS	2016-09-14	100.00	24.00	11284.00
8	1003	103	Lanturi Auto	2016-09-14	50.00	12.00	5642.00
9	1003	103	Scaun Copil Auto	2016-09-14	50.00	12.00	5642.00
10	1004	104	Centura Siguranta Auto	2015-05-24	200.00	48.00	7688.00
11	1005	105	Scaun Copil Auto	2017-01-02	300.00	72.00	91884.00
12	1006	106	Autoturism	2016-08-29	200.00	48.00	6448.00
13	1006	106	Carut Copil	2016-08-29	50.00	12.00	1612.00
14	1007	107	Centura Siguranta Auto	2016-02-14	200.00	48.00	3968.00
15	1007	107	GPS	2016-02-14	50.00	12.00	992.00
16	1007	107	Lanturi Auto	2016-02-14	50.00	12.00	992.00
17	1008	108	Autoturism	2016-06-18	200.00	48.00	10416.00
18	1008	108	Carut Copil	2016-06-18	50.00	12.00	2604.00
19	1008	108	Plasa separatoare Animal	2016-06-18	30.00	7.20	1562.40
20	1009	109	Autoturism	2016-11-09	100.00	24.00	4836.00
21	1009	109	Centura Siguranta Auto	2016-11-09	30.00	7.20	1450.80
22	1009	109	GPS	2016-11-09	50.00	12.00	2418.00
23	1009	109	Lanturi Auto	2016-11-09	30.00	7.20	1450.80
24			Ochelari de Condus Night View				

R:



	idcontract	nrfactură	datafacturii	denprod	VALOARE_INCHIRIERI_PER_ZI	VALOARE_TVA_PER_ZI	VALOARE_TOTALA
1	1000	100	2017-05-24	Autoturism	200	48.0	22816.0
2	1000	100	2017-05-24	GPS	50	12.0	5704.0
3	1000	100	2017-05-24	Scaun Copil Auto	100	24.0	11408.0
4	1000	100	2017-05-24	Plasa separatoare Animal	100	24.0	11408.0
5	1001	101	2017-02-12	Carut Copil	300	72.0	56544.0
6	1002	102	2016-07-13	Lanturi Auto	400	96.0	24800.0
7	1003	103	2016-09-14	GPS	100	24.0	11284.0
8	1003	103	2016-09-14	Scaun Copil Auto	50	12.0	5642.0
9	1003	103	2016-09-14	Lanturi Auto	50	12.0	5642.0
10	1004	104	2019-05-24	Centura Siguranta Auto	200	48.0	7688.0
11	1005	105	2017-01-02	Scaun Copil Auto	300	72.0	91884.0
12	1006	106	2016-08-29	Autoturism	200	48.0	6448.0
13	1006	106	2016-08-29	Carut Copil	50	12.0	1612.0
14	1007	107	2016-02-14	GPS	50	12.0	992.0
15	1007	107	2016-02-14	Lanturi Auto	50	12.0	992.0
16	1007	107	2016-02-14	Centura Siguranta Auto	200	48.0	3968.0
17	1008	108	2016-06-18	Autoturism	200	48.0	10416.0
18	1008	108	2016-06-18	Carut Copil	50	12.0	2604.0
19	1008	108	2016-06-18	Plasa separatoare Animal	30	7.2	1562.4
20	1009	109	2016-11-09	Autoturism	100	24.0	4836.0
21	1009	109	2016-11-09	GPS	50	12.0	2418.0
22	1009	109	2016-11-09	Lanturi Auto	30	7.2	1450.8
23	1009	109	2016-11-09	Centura Siguranta Auto	30	7.2	1450.8
24	NA	NA	NA	Ochelari de Condus Night View	NA	NA	NA

Showing 1 to 24 of 24 entries

Figura 5. Rezultatul joncțiunii externe la stânga LEFT JOIN a patru table în SQL și R.

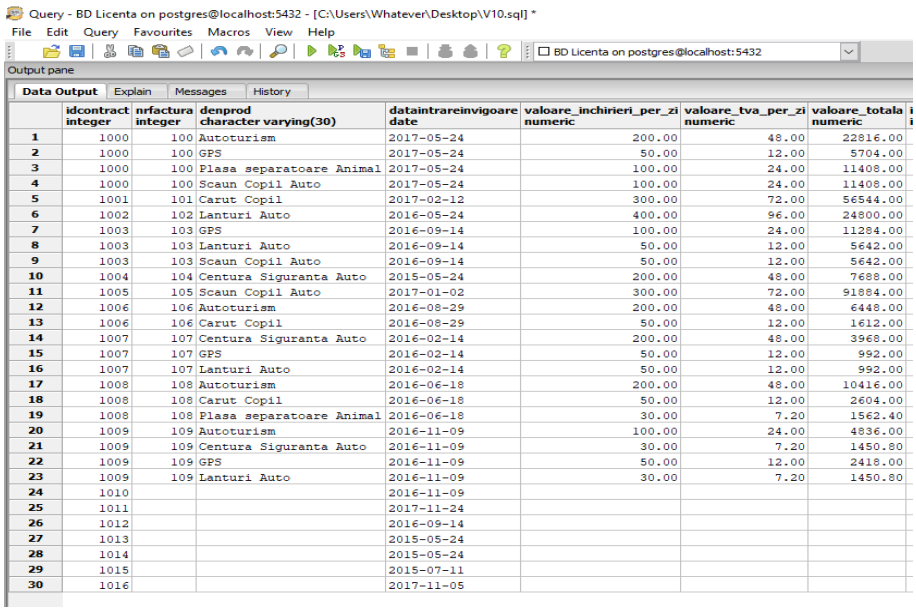
R:

	codprod	denprod	um	grupă	proctvă	nrfactură	liniefactură	cantitate	pretunit	tvalinieпr	idcontract	datafacturii	idclient	obs	valoarefactură	valoa
1	2007	Ochelari de Condus Night View	buc	NA	0.24	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Figura 6. Rezultatul joncțiunii anti_join() dintre tabela din R.

Joncțiunea externă la dreapta (RIGHT OUTER JOIN)

SQL:



Query - BD Licenta on postgres@localhost:5432 - [C:\Users\Whatever\Desktop\V10.sql] *

File Edit Query Favourites Macros View Help

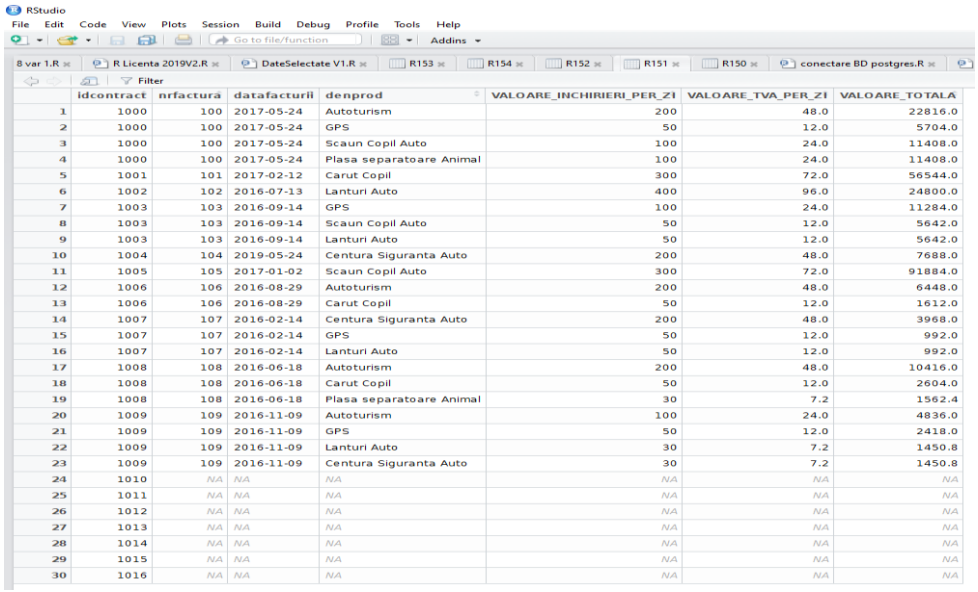
BD Licenta on postgres@localhost:5432

Output pane

Data Output | Explain | Messages | History

	idcontract integer	nrfactura integer	denprod character varying(30)	dataintrareinvigoare date	valoare_inchirieri_per_zi numeric	valoare_tva_per_zi numeric	valoare_totala numeric
1	1000	100	Autoturism	2017-05-24	200.00	48.00	22816.00
2	1000	100	GPS	2017-05-24	50.00	12.00	5704.00
3	1000	100	Plasa separatoare Animal	2017-05-24	100.00	24.00	11408.00
4	1000	100	Scaun Copil Auto	2017-05-24	100.00	24.00	11408.00
5	1001	101	Carut Copil	2017-02-12	300.00	72.00	56544.00
6	1002	102	Lanturi Auto	2016-05-24	400.00	96.00	24800.00
7	1003	103	GPS	2016-09-14	100.00	24.00	11284.00
8	1003	103	Lanturi Auto	2016-09-14	50.00	12.00	5642.00
9	1003	103	Scaun Copil Auto	2016-09-14	50.00	12.00	5642.00
10	1004	104	Centura Siguranta Auto	2015-05-24	200.00	48.00	7688.00
11	1005	105	Scaun Copil Auto	2017-01-02	300.00	72.00	91884.00
12	1006	106	Autoturism	2016-08-29	200.00	48.00	6448.00
13	1006	106	Carut Copil	2016-08-29	50.00	12.00	1612.00
14	1007	107	Centura Siguranta Auto	2016-02-14	200.00	48.00	3968.00
15	1007	107	GPS	2016-02-14	50.00	12.00	992.00
16	1007	107	Lanturi Auto	2016-02-14	50.00	12.00	992.00
17	1008	108	Autoturism	2016-06-18	200.00	48.00	10416.00
18	1008	108	Carut Copil	2016-06-18	50.00	12.00	2604.00
19	1008	108	Plasa separatoare Animal	2016-06-18	30.00	7.20	1562.40
20	1009	109	Autoturism	2016-11-09	100.00	24.00	4836.00
21	1009	109	Centura Siguranta Auto	2016-11-09	30.00	7.20	1450.80
22	1009	109	GPS	2016-11-09	50.00	12.00	2418.00
23	1009	109	Lanturi Auto	2016-11-09	30.00	7.20	1450.80
24	1010			2016-11-09			
25	1011			2017-11-24			
26	1012			2016-09-14			
27	1013			2015-05-24			
28	1014			2015-05-24			
29	1015			2015-07-11			
30	1016			2017-11-05			

R:



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

8 var 1.R | R Licenta 2019V2.R | DateSelectate V1.R | R153 | R154 | R152 | R151 | R150 | conectare BD postgres.R

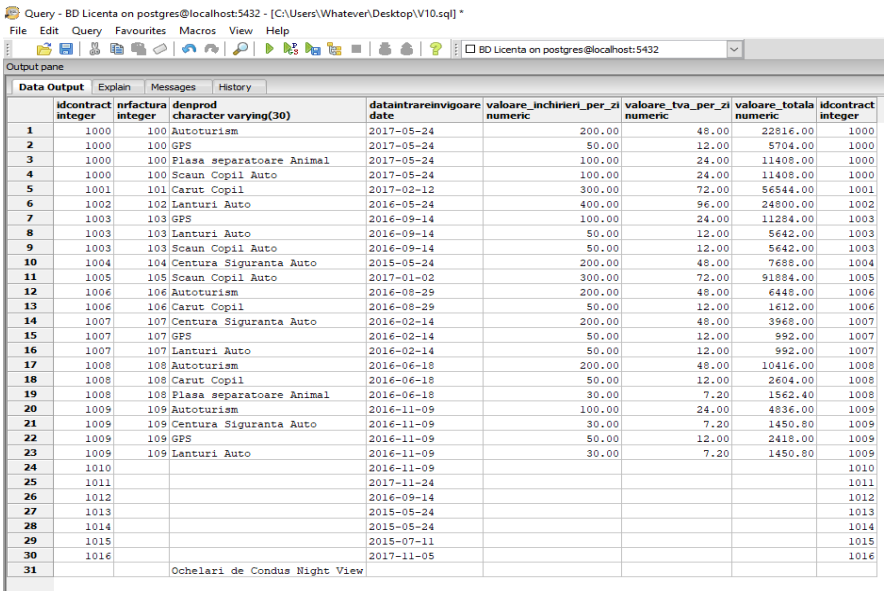
Filter

	idcontract	nrfactură	datafacturii	denprod	VALOARE_INCHIRIERI_PER_ZI	VALOARE_TVA_PER_ZI	VALOARE_TOTALĂ
1	1000	100	2017-05-24	Autoturism	200	48.0	22816.0
2	1000	100	2017-05-24	GPS	50	12.0	5704.0
3	1000	100	2017-05-24	Scaun Copil Auto	100	24.0	11408.0
4	1000	100	2017-05-24	Plasa separatoare Animal	100	24.0	11408.0
5	1001	101	2017-02-12	Carut Copil	300	72.0	56544.0
6	1002	102	2016-07-13	Lanturi Auto	400	96.0	24800.0
7	1003	103	2016-09-14	GPS	100	24.0	11284.0
8	1003	103	2016-09-14	Scaun Copil Auto	50	12.0	5642.0
9	1003	103	2016-09-14	Lanturi Auto	50	12.0	5642.0
10	1004	104	2019-05-24	Centura Siguranta Auto	200	48.0	7688.0
11	1005	105	2017-01-02	Scaun Copil Auto	300	72.0	91884.0
12	1006	106	2016-08-29	Autoturism	200	48.0	6448.0
13	1006	106	2016-08-29	Carut Copil	50	12.0	1612.0
14	1007	107	2016-02-14	Centura Siguranta Auto	200	48.0	3968.0
15	1007	107	2016-02-14	GPS	50	12.0	992.0
16	1007	107	2016-02-14	Lanturi Auto	50	12.0	992.0
17	1008	108	2016-06-18	Autoturism	200	48.0	10416.0
18	1008	108	2016-06-18	Carut Copil	50	12.0	2604.0
19	1008	108	2016-06-18	Plasa separatoare Animal	30	7.2	1562.4
20	1009	109	2016-11-09	Autoturism	100	24.0	4836.0
21	1009	109	2016-11-09	GPS	50	12.0	2418.0
22	1009	109	2016-11-09	Lanturi Auto	30	7.2	1450.8
23	1009	109	2016-11-09	Centura Siguranta Auto	30	7.2	1450.8
24	1010	NA	NA	NA	NA	NA	NA
25	1011	NA	NA	NA	NA	NA	NA
26	1012	NA	NA	NA	NA	NA	NA
27	1013	NA	NA	NA	NA	NA	NA
28	1014	NA	NA	NA	NA	NA	NA
29	1015	NA	NA	NA	NA	NA	NA
30	1016	NA	NA	NA	NA	NA	NA

Figura 7. Rezultatul joncțiunii externe la dreapta RIGHT JOIN a patru table în SQL și R.

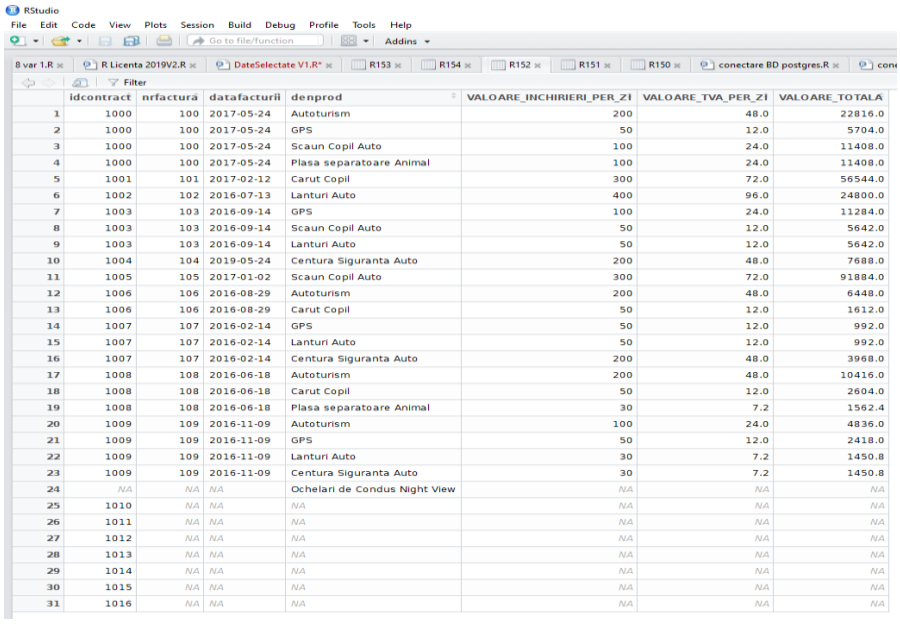
Joncțiunea externă (FULL OUTER JOIN)

SQL:



	idcontract integer	nrfactura integer	denprod character varying(30)	dataintrareinvigoare date	valoare_inchirieri_per_zi numeric	valoare_tva_per_zi numeric	valoare_totala numeric	idcontract integer
1	1000	100	Autoturism	2017-05-24	200.00	48.00	22816.00	1000
2	1000	100	GPS	2017-05-24	50.00	12.00	5704.00	1000
3	1000	100	Plasa separatoare Animal	2017-05-24	100.00	24.00	11408.00	1000
4	1000	100	Scaun Copil Auto	2017-05-24	100.00	24.00	11408.00	1000
5	1001	101	Carut Copil	2017-02-12	300.00	72.00	56544.00	1001
6	1002	102	Lanturi Auto	2016-05-24	400.00	96.00	24800.00	1002
7	1003	103	GPS	2016-09-14	100.00	24.00	11284.00	1003
8	1003	103	Lanturi Auto	2016-09-14	50.00	12.00	5642.00	1003
9	1003	103	Scaun Copil Auto	2016-09-14	50.00	12.00	5642.00	1003
10	1004	104	Centura Siguranta Auto	2015-05-24	200.00	48.00	7688.00	1004
11	1005	105	Scaun Copil Auto	2017-01-02	300.00	72.00	91884.00	1005
12	1006	106	Autoturism	2016-08-29	200.00	48.00	6448.00	1006
13	1006	106	Carut Copil	2016-08-29	50.00	12.00	1612.00	1006
14	1007	107	Centura Siguranta Auto	2016-02-14	200.00	48.00	3968.00	1007
15	1007	107	GPS	2016-02-14	50.00	12.00	992.00	1007
16	1007	107	Lanturi Auto	2016-02-14	50.00	12.00	992.00	1007
17	1008	108	Autoturism	2016-06-18	200.00	48.00	10416.00	1008
18	1008	108	Carut Copil	2016-06-18	50.00	12.00	2604.00	1008
19	1008	108	Plasa separatoare Animal	2016-06-18	30.00	7.20	1562.40	1008
20	1009	109	Autoturism	2016-11-09	100.00	24.00	4836.00	1009
21	1009	109	Centura Siguranta Auto	2016-11-09	30.00	7.20	1450.80	1009
22	1009	109	GPS	2016-11-09	50.00	12.00	2418.00	1009
23	1009	109	Lanturi Auto	2016-11-09	30.00	7.20	1450.80	1009
24	1010			2016-11-09				1010
25	1011			2017-11-24				1011
26	1012			2016-09-14				1012
27	1013			2015-05-24				1013
28	1014			2015-05-24				1014
29	1015			2015-07-11				1015
30	1016			2017-11-05				1016
31			Ochelari de Condus Night View					

R:



	idcontract	nrfactură	datafacturii	denprod	VALOARE_INCHIRIERI_PER_ZI	VALOARE_TVA_PER_ZI	VALOARE_TOTALĂ
1	1000	100	2017-05-24	Autoturism	200	48.0	22816.0
2	1000	100	2017-05-24	GPS	50	12.0	5704.0
3	1000	100	2017-05-24	Scaun Copil Auto	100	24.0	11408.0
4	1000	100	2017-05-24	Plasa separatoare Animal	100	24.0	11408.0
5	1001	101	2017-02-12	Carut Copil	300	72.0	56544.0
6	1002	102	2016-07-13	Lanturi Auto	400	96.0	24800.0
7	1003	103	2016-09-14	GPS	100	24.0	11284.0
8	1003	103	2016-09-14	Scaun Copil Auto	50	12.0	5642.0
9	1003	103	2016-09-14	Lanturi Auto	50	12.0	5642.0
10	1004	104	2019-05-24	Centura Siguranta Auto	200	48.0	7688.0
11	1005	105	2017-01-02	Scaun Copil Auto	300	72.0	91884.0
12	1006	106	2016-08-29	Autoturism	200	48.0	6448.0
13	1006	106	2016-08-29	Carut Copil	50	12.0	1612.0
14	1007	107	2016-02-14	GPS	50	12.0	992.0
15	1007	107	2016-02-14	Lanturi Auto	50	12.0	992.0
16	1007	107	2016-02-14	Centura Siguranta Auto	200	48.0	3968.0
17	1008	108	2016-06-18	Autoturism	200	48.0	10416.0
18	1008	108	2016-06-18	Carut Copil	50	12.0	2604.0
19	1008	108	2016-06-18	Plasa separatoare Animal	30	7.2	1562.4
20	1009	109	2016-11-09	Autoturism	100	24.0	4836.0
21	1009	109	2016-11-09	GPS	50	12.0	2418.0
22	1009	109	2016-11-09	Lanturi Auto	30	7.2	1450.8
23	1009	109	2016-11-09	Centura Siguranta Auto	30	7.2	1450.8
24	NA	NA	NA	Ochelari de Condus Night View	NA	NA	NA
25	1010	NA	NA	NA	NA	NA	NA
26	1011	NA	NA	NA	NA	NA	NA
27	1012	NA	NA	NA	NA	NA	NA
28	1013	NA	NA	NA	NA	NA	NA
29	1014	NA	NA	NA	NA	NA	NA
30	1015	NA	NA	NA	NA	NA	NA
31	1016	NA	NA	NA	NA	NA	NA

Figura 8. Rezultatul joncțiunii externe JULL JOIN a patru table în SQL și R.

Grupări

Grupari simple

SQL:

Query - BD Licenta on postgres@localhost:543

File Edit Query Favourites Macros View

Output pane

Data Output | Explain | Messages | History

	nrfactura integer	TOTAL incasari cu TVA numeric
1	100	4808.56
2	101	2071.38
3	102	3777.34
4	103	13532.82
5	104	1123.34
6	105	606.14
7	106	4722.72
8	107	4869.21
9	108	6810.42
10	109	6650.16

R:

RStudio

File Edit Code View Plots Session Build Debug

8 var 1.R | R Licenta 2019V2.R | DateSelectate

Filter

	nrfactura	Total_incasari_cu_TVA
1	100	4808.56
2	101	2071.38
3	102	3777.34
4	103	13532.82
5	104	1123.34
6	105	606.14
7	106	4722.72
8	107	4869.21
9	108	6810.42
10	109	6650.16

Figura 9. Lista facturilor și valoarea acestora în SQL și R.

SQL:

Query - BD Licenta on postgres@localhost:5432 - [C:\Users\Whatever\Desktop\V

File Edit Query Favourites Macros View Help

Output pane

Data Output | Explain | Messages | History

	numeclient character varying(50)	date_part double precision	valoare_totala numeric	nr_facturi bigint
1	APOSTOL LARISA	8	61395.36	1
2	ARDELEANU ION	6	95345.88	1
3	AUGUSTIN ABBOTT	2	25969.12	1
4	CATE HAMILTON	11	64839.06	1
5	MATEI CARMEN	7	188867.00	1
6	VASILE ANDREI	9	410495.54	1

R:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Addins

8 var 1.R | R Licenta 2019V2.R | DateSelectate V1.R* | R153 | R154

Filter

	numeclient	month(datafacturii)	valoare_serviciu	nr_facturi
1	APOSTOL LARISA	8	61395.36	1
2	ARDELEANU ION	6	95345.88	1
3	AUGUSTIN ABBOTT	2	25969.12	1
4	CATE HAMILTON	11	64839.06	1
5	MATEI CARMEN	7	188867.00	1
6	VASILE ANDREI	9	410495.54	1

Figura 10. Lista clienților și valoarea serviciilor facturate în anul 2016 în SQL și R.

SQL:

Query - BD Licenta on postgres@localhost:5432 - [C:\Users

File Edit Query Favourites Macros View Help

Output pane

Data Output | Explain | Messages | History

	nrfactura integer	TOTAL incasari cu TVA numeric
1	103	13532.82
2	108	6810.42
3	109	6650.16

R:

RStudio

File Edit Code View Plots Session Build Debug Profile

8 var 1.R | R Licenta 2019V2.R | DateSelectate V1.R*

Filter

	nrfactura	Total_chirie_per_zi
1	103	13532.82
2	108	6810.42
3	109	6650.16

Figura 11. Lista clienților a căror valoarea a chiriei plătite pe zi este mai mare de 6000 lei în SQL și R.

SQL:

Query - BD Licenta on postgres@localhost:5432 - [C:\Users\Whatever\D

File Edit Query Favourites Macros View Help

Output pane

Data Output | Explain | Messages | History

	numebirou character varying	date_part double precision	valoare_serviciu numeric
1	Birou 1	2016	48087.20
2	Birou 1	2017	96472.00
3	Birou 1	2019	7688.00
4	Birou 1 - SUBTOTAL		165106.00
5	Birou 2	2016	21886.00
6	Birou 2 - SUBTOTAL		28520.00
7	Birou 3	2016	8060.00
8	Birou 3	2017	91884.00
9	Birou 3 - SUBTOTAL		99944.00

R:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins

8 var 1.R | R Licenta 2019V2.R | DateSelectate V1.R* | R153 | R1

Filter

	category	numebirou	year(datafacturii)	valoare_serviciu
1	1	Birou 1	2016	48087.2
2	1	Birou 1	2017	96472.0
3	1	Birou 1	2019	7688.0
4	1	Total	NA	165106.0
5	2	Birou 2	2016	21886.0
6	2	Total	NA	28520.0
7	3	Birou 3	2016	8060.0
8	3	Birou 3	2017	91884.0
9	3	Total	NA	99944.0

Figura 12. Raport privind câștigurile realizate de fiecare birou în SQL și R.

SQL:

Query - BD Licenta on postgres@localhost:5432 - [C:\Users\Whatever\Desktop\V10.sql] *

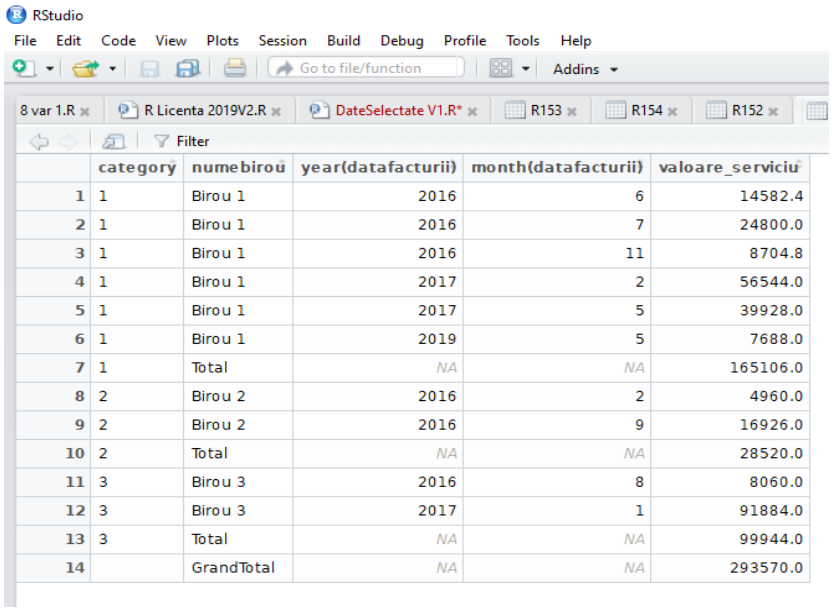
File Edit Query Favourites Macros View Help

Output pane

Data Output | Explain | Messages | History

	numebirou character varying	date_part double precision	date_part double precision	valoare_serviciu numeric
1	Birou 1	2016	6	14582.40
2	Birou 1	2016	7	24800.00
3	Birou 1	2016	11	10155.60
4	Birou 1	2017	2	56544.00
5	Birou 1	2017	5	51336.00
6	Birou 1	2019	5	7688.00
7	Birou 1 - SUBTOTAL			165106.00
8	Birou 2	2016	2	5952.00
9	Birou 2	2016	9	22568.00
10	Birou 2 - SUBTOTAL			28520.00
11	Birou 3	2016	8	8060.00
12	Birou 3	2017	1	91884.00
13	Birou 3 - SUBTOTAL			99944.00
14	TOTAL GENERAL			293570.00

R:



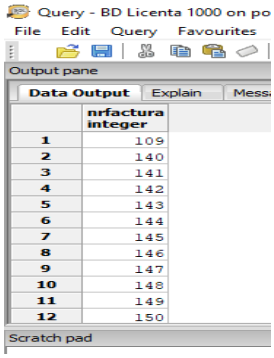
	categorý	numebirou	year(datafacturii)	month(datafacturii)	valoare_serviciu
1	1	Birou 1	2016	6	14582.4
2	1	Birou 1	2016	7	24800.0
3	1	Birou 1	2016	11	8704.8
4	1	Birou 1	2017	2	56544.0
5	1	Birou 1	2017	5	39928.0
6	1	Birou 1	2019	5	7688.0
7	1	Total	NA	NA	165106.0
8	2	Birou 2	2016	2	4960.0
9	2	Birou 2	2016	9	16926.0
10	2	Total	NA	NA	28520.0
11	3	Birou 3	2016	8	8060.0
12	3	Birou 3	2017	1	91884.0
13	3	Total	NA	NA	99944.0
14		GrandTotal	NA	NA	293570.0

Figura 13. Raport privind câștigurile realizate pe fiecare birou și la nivel de firmă în SQL și R.

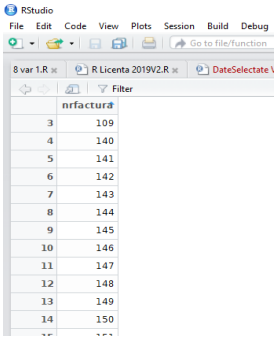
3.4. Subinterogări

Subinterogări în clauza WHERE

SQL: **R:**



	nrfactura integer
1	109
2	140
3	141
4	142
5	143
6	144
7	145
8	146
9	147
10	148
11	149
12	150



	nrfactura
3	109
4	140
5	141
6	142
7	143
8	144
9	145
10	146
11	147
12	148
13	149
14	150

Figura 14. Lista facturilor emise în aceeași zi cu factura numărul 209 în SQL și R.

Subinterogări în clauza HAVING

SQL: **R:**

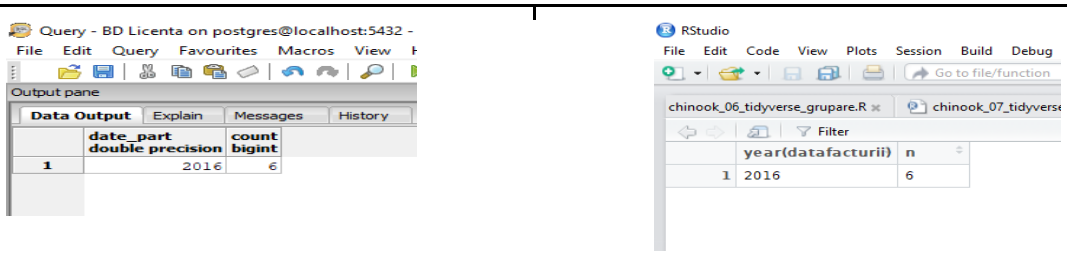


Figura 15. Anul în care au fost emise cele mai multe facturi în SQL și R.

Subinterogări în clauza FROM

SQL: **R:**

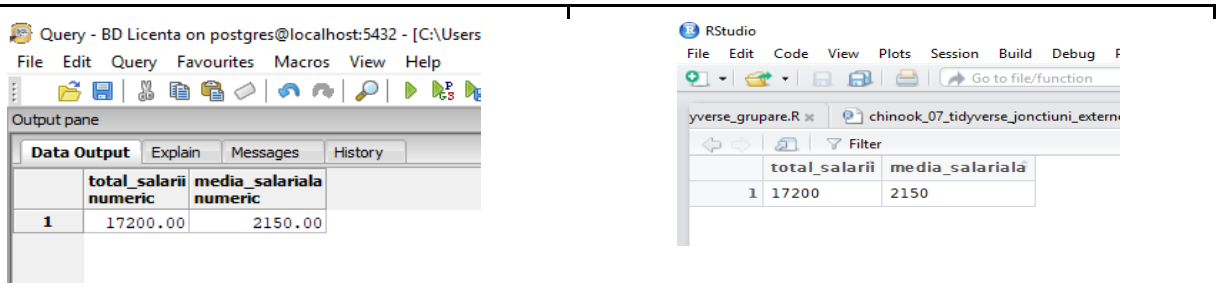


Figura 16. Calculul valorii medii a salariului și cheltuielile salariale la nivel de firmă în SQL și R.

SQL: **R:**

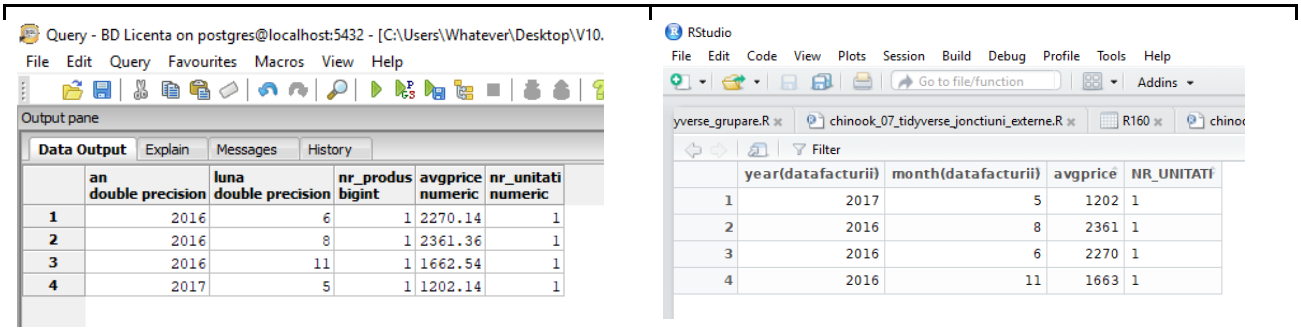


Figura 17. Prețul mediu de închiriere a serviciilor pe an și pe lună în SQL și R.

Subinterogări în clauza SELECT

SQL: **R:**

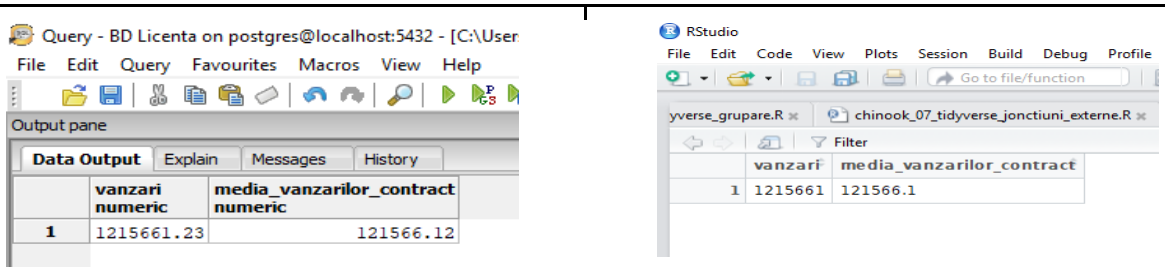


Figura 18. Valoarea vânzărilor totale și valoarea medie a contractelor în SQL și R.

3.5. Diferențe, reuniuni și intersecții

SQL: **R:**

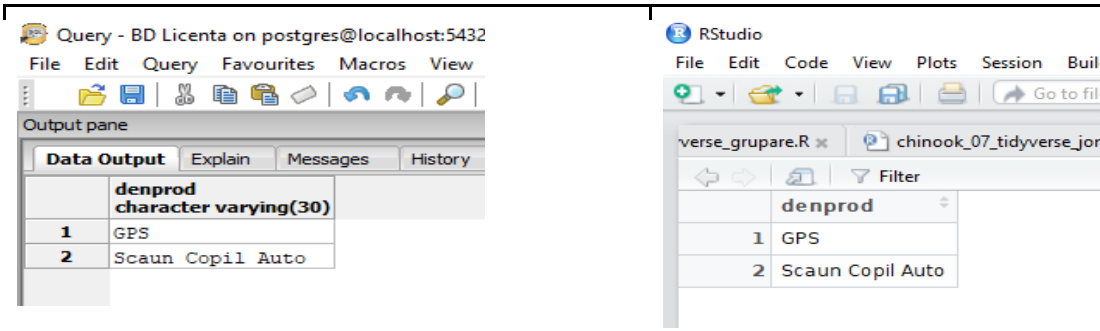


Figura 18. Lista produselor care nu se regăsesc pe factura numărul 108 în SQL și R.

SQL: **R:**

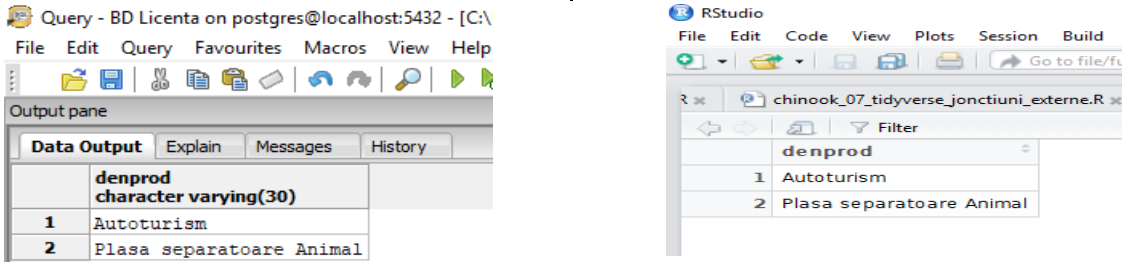
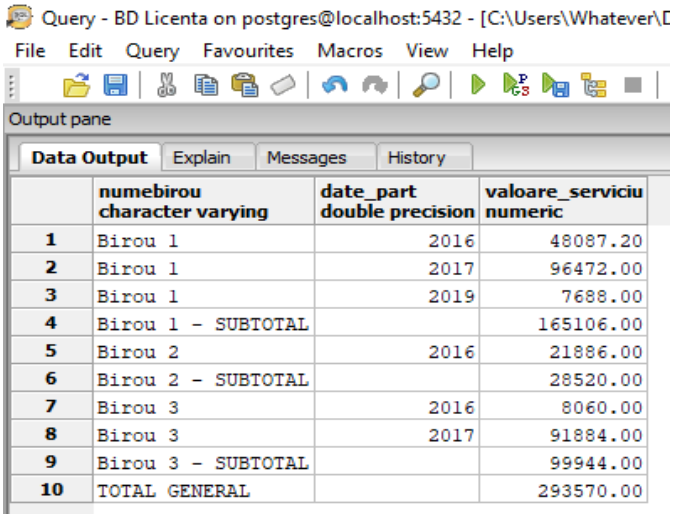


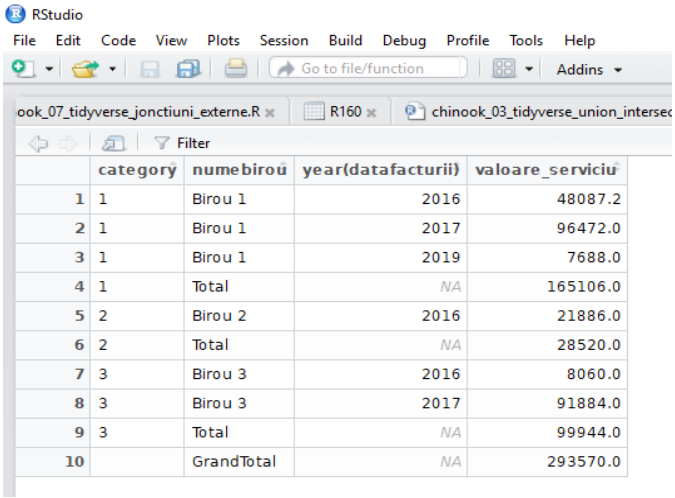
Figura 20. Lista produselor comune regăsite pe două facturi.

SQL:



	numebirou character varying	date_part double precision	valoare_serviciu numeric
1	Birou 1	2016	48087.20
2	Birou 1	2017	96472.00
3	Birou 1	2019	7688.00
4	Birou 1 - SUBTOTAL		165106.00
5	Birou 2	2016	21886.00
6	Birou 2 - SUBTOTAL		28520.00
7	Birou 3	2016	8060.00
8	Birou 3	2017	91884.00
9	Birou 3 - SUBTOTAL		99944.00
10	TOTAL GENERAL		293570.00

R:

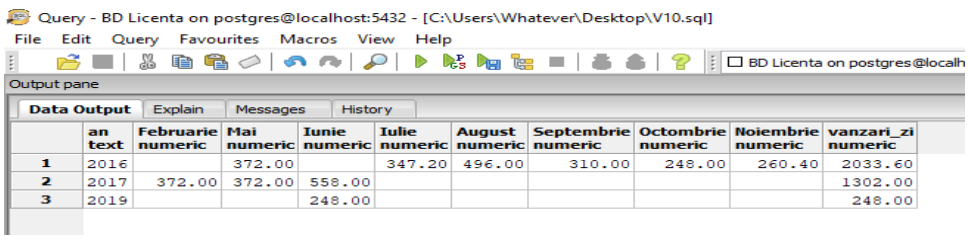


	category	numebirou	year(datafacturii)	valoare_serviciu
1	1	Birou 1	2016	48087.2
2	1	Birou 1	2017	96472.0
3	1	Birou 1	2019	7688.0
4	1	Total	NA	165106.0
5	2	Birou 2	2016	21886.0
6	2	Total	NA	28520.0
7	3	Birou 3	2016	8060.0
8	3	Birou 3	2017	91884.0
9	3	Total	NA	99944.0
10		GrandTotal	NA	293570.0

Figura 21. Lista vânzărilor pe birou cu subtotaluri pe fiecare birou și un total general în SQL și R.

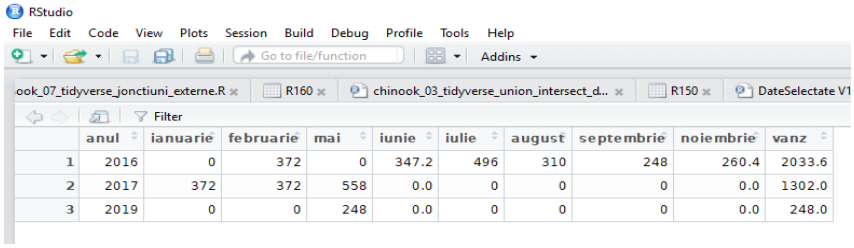
3.6. Tabele pivot

SQL:



	an text	Februarie numeric	Mai numeric	Iunie numeric	Iulie numeric	August numeric	Septembrie numeric	Octombrie numeric	Noiembrie numeric	vanzari_zi numeric
1	2016		372.00		347.20	496.00	310.00	248.00	260.40	2033.60
2	2017	372.00	372.00	558.00						1302.00
3	2019			248.00						248.00

R:

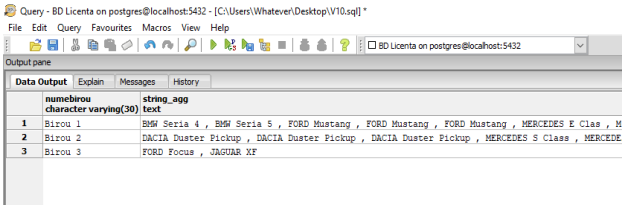


	anul	ianuarie	februarie	mai	iunie	iulie	august	septembrie	noiembrie	vanz
1	2016	0	372	0	347.2	496	310	248	260.4	2033.6
2	2017	372	372	558	0.0	0	0	0	0.0	1302.0
3	2019	0	0	248	0.0	0	0	0	0.0	248.0

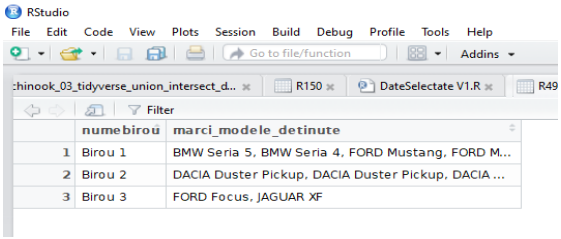
Figura 22. Raportul vânzărilor pe zi a produselor închiriate în perioada 2016 - 2019 în SQL și R.

3.7. Recursivitate

SQL:



R:



	numebirou	marci_modele_detinute
1	Birou 1	BMW Seria 5, BMW Seria 4, FORD Mustang, FORD M...
2	Birou 2	DACIA Duster Pickup, DACIA Duster Pickup, DACIA ...
3	Birou 3	FORD Focus, JAGUAR XF

Figura 20. Lista produselor comune regăsite pe două facturi.

Bibliografie

Carți:

1. Antoniou, G., Papoglou, N., Business Intelligence & Analytics (BI&A) Systems Measuring End-User Computing Satisfaction (EUCS), Disertatie, Universitatea Lund, 2015.
2. Conway, J., Eddelbuettel, D., Nishiyama, T., Prayaga, S., K.,Tiffin, N., RPostgreSQL: R Interface to the 'PostgreSQL' Database System, 2017, disponibil la https://cran.r-project.org/web/packages/RPostgreSQL/ index.html.
3. Dusa, A., Oancea, B., Caragea, N., Alexandru, C., Jula, N., M., Dobre, A., M., R cu aplicații in statistica, Ed. Universității din București, 2015.
4. Eckerson, W., Smart Companies in the 21st Century: The Secrets of Creating Successful Business Intelligence Solutions, The Data Warehousing Institute, Seattle, 2003.
5. Fotache, M, SQL, Dialecte BD2, Oracle, PostgreSQL si SQL Server editia a II-a, Ed. Polirom, 2009.
6. Han, J., Kamber, M., Pei, J., Data mining: Concepts and Techniques, Editia a III-a, Morgan Kaufmann Publisher, San Francisco, 2012.
7. Kabacoff, R., I., R in Action. Data analysis and graphics with R, Manning Publications Co., New York, 2011.
8. Kamber, M., Data mining: Concepts and Techniques Editia II, Morgan Kaufmann Publisher, San Francisco, 2006.
9. Kilin, S., Review of modern business intelligence and analytics in 2015: How to tame the big data in practice/Case study – What kind of modern business intelligence and analytics strategy to choose?, disertatie, disponibila la https://aaltodoc.aalto.fi/bitstream/handle/123456789/18349/master_Kulin_Samu_2015.pdf?sequence=1&isAllowed=y, 2015.
10. Maksimov, D., Performance Comparison of MongoDB and PostgreSQL with JSON types, Tallinn University of technology, Faculty of Information Technology Department of Informatics, Disertatie, 2015.
11. Nagy, I., M., Proiectarea și Implementarea Depozitelor de Date pentru Business Intelligence aplicate în Economie, Teza de doctorat - rezumat, Cluj-Napoca, 2012.
12. Raisinghani, M., Business Intelligence in the Digital Economy: Opportunities, Limitations and Risks, Idea Group Inc, University of Dallas, USA, 2004.
13. Turnban, E., Aronson, J., E., Liang, T., P., Decision support systems and intelligent systems, Editia a 7-a, Pearson Education, Inc., New Jersey, 2007.
14. Wickham, H., Grolemund, G., R for data science. Import, Tidy, transform, visualize and model data, Published by O'Reilly Media, Inc., Canada, 2016.
15. Williams, S., Business Intelligence Strategy and Big Data Analytics. A General Management Perspective, Morgan Kaufmann Cambridge, USA, 2016.

Articole științifice:

16. Alazemi, A., R., Data, text, and web mining for business intelligence: a survey, International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.3, No.2, 2013.
17. Anuradha, G., Joel Varma, D., Graph Mining Extensions in Postgresql, Indian Journal of Science and Technology, Vol 9(35), 2016.
18. Chaudhuri, S, Dayal, U., Narasayya,V., An Overview of Business Intelligence Technology, Communications of the ACM, Vol. 54 No. 8, p. 88-98, 2011.

19. Chen, H., Chiang, R., H., L., Storey, V., C., Business Intelligence and Analytics: from big data to big impact, MIS Quarterly, Vol. 36, nr. 4, pp. 1165-1188, 2012.
20. Coardos, D., Marinescu, I., A., Soluții de tip BI pentru asistarea deciziilor în administrarea publica locala, Revista Română de Informatică și Automatică, vol. 25, nr. 2, 2015.
21. Fotache, M, Strimbei, C., SQL and data analysis. Some implications for data analysits and higher education, Procedia Economics and Finance 20, 2015.
22. Fotache, M., Data Processing Languages for Business Intelligence. SQL vs. R, Informatica Economică vol. 20, nr. 1, 2016.
23. Gang-Hoon, K., Trimi, S., Ji-Hyong, C., Big Data Applications in the Government Sector: A Comparative Analysis among Leading Countries, Communications of the ACM, vol. 57, nr. 3, 2014.
24. Habimana, J., Query Optimization Techniques - Tips For Writing Efficient And Faster SQL Queries, International journal of scientific & technology research, 4, 10, 2015.
25. Hrubaru, I., Fotache, M., On the Performance of Three In-Memory Data Systems for On Line Analytical Processing, Informatica Economica vol. 21, no. 1, 2017.
26. Mazareanu, V., The Intelligence in Business Intelligence, Analele Științifice ale Universității "Alexandru Ioan Cuza", Științe Economice, Iași, 2006.
27. Mohammed, J., Business Intelligence and Analytics Evolution, Applications, and Emerging Research Areas, International Journal of Engineering Science and Innovative Technology (IJESIT) Vol. 4, 2, 2015.
28. Muntean, M., Surcel, T., Agile BI – The Future of BI, Informatica Economică vol. 17, nr. 3, 2013.
29. Silva, Y., N., Almeida, I., Queiroz, M., SQL: From traditional database to Big Data, SIGCSE'16, Memphis, Tennessee, USA, 2016.
30. Sivarajah, U., Kamal, M., Vishanth, Z.,Critical analysis of Big Data challenges and analytical methods, Journal of Business Research, vol. 70, 2017.

Articole web:

31. Bashir, S., Getting started in R – Tidyverse Edition, 2019, http://ilustat.com/shared/ Getting-Started-in-R.pdf.
32. Caragea, N., Alexandru, C., A., Dobre, A., M., Bringing new opportunities to develop statistical software and data analysis tools in Romania, MPRA nr. 48772, 2013.
33. Champagne, J., The Top 7 Free and Open Source Database Software Solutions, 2017, disponibil la https://blog.capterra.com/ free-database-software/.
34. Drake, M., SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems, 2019, disponibil la https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of- relational-database-management-systems.
35. Erhardt, E., B., Introduction into R tidyverse, prezentare, 2018, https://statacumen.com/teach/ShortCourses/R_Programming/IntroTidyverse_ACASA_201802/Erhardt_RTidyverse_20180216.pdf.
36. Gaille, B., 14 Pros and Cons of Business Intelligence, 2016, https://brandongaille.com/14-pros-and-cons -of-business-intelligence/, accesat la 27.02.2019.Herschel, G., Linden, A., Kart, L., Magic Quadrant for Advanced Analytics Platforms, 2015, http://www.cscbrasil.com.br/alteryx/wp-content/uploads/2015/05/Magic- Quadrant-for-Advanced-Analytics-Platforms.pdf.
37. Mainmone, C., R: Working with Databases, 2018, https://nuitrcs.github.io/databases_workshop/r/r_ databases.html#execute-queries.

38. O'Reilly, Data Science Salary Survey results, 2016, disponibil la https://blog.revolutionanalytics.com/popularity/page/2/, accesat la 12.03.2019.
39. Peng, R., D., R Programming for Data Science, 2015, https://www.cs.upc.edu/~robert/teaching/estadistica/ rprogramming.pdf.
40. Piatetsky, G., New Leader, Trends, and Surprises in Analytics, Data Science, Machine Learning Software Poll, 2017, disponibil la https://www.kdnuggets.com/2017/05/poll - analytics-data-science-machine-learning -software-leaders.html.
41. Porzak, J., Data Wrangling in the Tidyverse 21st Century R, prezentare, 2017, https://ds4ci.files.wordpress.com /2017/04/ds-portugal-april-2017.pdf.
42. Stonebraker, M., What Does 'big Data' Mean?, 2012, disponibil la https://cacm.acm.org/blogs/blog-cacm/155468 -what-does-big-data-mean/fulltext.
43. Walker, A., 18 Best Open-Source and Free Database Software, 2017, disponibil la https://learn.g2crowd.com/free-database-software.
44. Wickham, H., Package 'tibble', 2019, https://cran.r-project.org/web/packages/tibble/tibble.pdf.

Articole web fara autor:

45. ***Count/tally observations by group, https://dplyr.tidyverse.org/reference/tally.html.
46. ***https://cloud.r-project.org/.
47. ***https://cran.r-project.org/bin/windows/base/.
48. ***https://data-flair.training/blogs/sql-tutorials-home/#interview-questions.
49. ***https://dplyr.tidyverse.org/.
50. ***https://forcats.tidyverse.org/.
51. ***https://mariadb.com/newsroom/press-releases/report-state-of-the-open-source-dbms-market-2018-by-gartner-includes -pricing-comparison-with-mariadb-2/.
52. ***https://purrr.tidyverse.org/.
53. ***https://readr.tidyverse.org/.
54. ***https://stat.ethz.ch/pipermail/r-announce/2018/000634.html.
55. ***https://stringr.tidyverse.org/.
56. ***https://tidyr.tidyverse.org/.
57. ***https://www.gnu.org/philosophy/philosophy.en.html.
58. ***https://www.postgresql.org/docs/9.1/tablefunc.html.
59. ***https://www.postgresql.org/docs/9.5/functions-aggregate.html.
60. ***https://www.splendiddata.com/2017/06/06/gartner-says-2018-70-new-applications-will-run-open-source-databases/.
61. ***Metrics Maven: Creating Pivot Tables in PostgreSQL Using Crosstab, 2016, https://www.compose.com/articles/metrics-maven-creating-pivot-tables-in-postgresql-using-crosstab/.
62. ***Open source software, https://www.onebusiness.ca/sites/default/files/MEDI_Booklet_Open_Source _Software_ accessible_E.pdf.
63. ***Overview, https://ggplot2.tidyverse.org/.
64. ***Performance comparison: odbc vs RPostgreSQL, https://rpubs.com/nwstephens/334324.
65. ***The "Tidyverse", http://www.araastat.com/BIOF339_PracticalR/Lectures/lecture_tidyverse.pdf.
66. ***The Pearson Correlation Coefficient Formula in SQL, https://chartio.com/learn/postgresql/correlation -coefficient-pearson/.
67. ***Tidyverse packages, https://www.tidyverse.org/packages/.