Osnove relacijskih baza podataka i SQL-a

Ishodi učenja

Nakon završetka ove nastavne cjeline studenti će moći:

- Objasniti temeljne pojmove relacijskih baza podataka i njihovu ulogu u upravljanju strukturiranim podacima.
- Opisati glavne sastavnice sustava baza podataka: tablice, veze, ključeve i ograničenja.
- Razumjeti načela normalizacije i primijeniti ih radi poboljšanja oblikovanja baze podataka.
- Pisati SQL upite primjenom naredbi SELECT, WHERE, GROUP BY, HAVING, JOIN, ORDER BY, INSERT, UPDATE i DELETE.
- Oblikovati osnovne relacijske sheme za podatke o okolišu (npr. praćenje kvalitete zraka i razine buke).
- Analizirati podatke o okolišu SQL-om (agregiranje, filtriranje, spajanje tablica i podupiti).
- Tumačiti rezultate upita i pripremiti kratka analitička izvješća.

Uvod

U digitalnim sustavima podaci su jedan od najvažnijih resursa. Agencije za zaštitu okoliša, istraživačke skupine i javne ustanove prikupljaju velike količine opažajnih podataka: atmosferska mjerenja, meteorološke podatke, pokazatelje kvalitete vode, inventare emisija i drugo. Bez sustavnog pristupa ti podaci postaju rascjepkani, nedosljedni i teško ih je analizirati.

Relacijske baze podataka organiziraju informacije u strukturirane tablice, pri čemu svaka tablica predstavlja određenu vrstu entiteta (npr. Stations, Measurements, Users).

Podatkovni model temelji se na relacijskom modelu koji je uveo E. F. Codd, a koji osigurava dosljednost i matematičku utemeljenost.

Glavna obilježja:

1. Tablična struktura

- Podaci se pohranjuju u tablicama (relacijama), koje se sastoje od redaka (n-torki) i stupaca (atributa).

- Svaka tablica ima definiranu shemu koja određuje nazive atributa, tipove podataka i ograničenja.

2. Strogo definirana shema

- Svi podaci moraju odgovarati unaprijed definiranoj strukturi.
- Svaki stupac ima definiranu domenu vrijednosti (cijeli broj, tekst, datum itd.).
- Ograničenja PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE i CHECK osiguravaju integritet podataka.

3. Strukturirani i dosljedni podaci

- Zapisi moraju u potpunosti odgovarati shemi. Primjerice, tablica AirQuality(station_id, datetime, pollutant, value) zahtijeva da svaki redak sadržava navedene atribute odgovarajućih tipova podataka.

4. Veze među tablicama

- Tablice se mogu međusobno povezivati ključevima, čime se omogućuje normalizirano oblikovanje i smanjuje redundancija podataka.

Zbog takve strukture relacijske baze podataka osobito su prikladne za jasno definirane, strukturirane podatke kod kojih je dosljednost presudna, primjerice u financijskim sustavima, znanstvenim skupovima podataka, upravljanju zalihama i praćenju stanja okoliša.

Pregled tehnologije

Temeljni pojmovi relacijskih baza podataka

Tablice i sheme

U relacijskom modelu svi se podaci organiziraju u logičke strukture koje nazivamo tablicama, odnosno formalno relacijama (tablica 1.). Tablica predstavlja jednu vrstu entiteta, primjerice mjernu postaju, mjerenje onečišćenja ili meteorološko opažanje, te omogućuje jednoobraznu pohranu više primjeraka toga entiteta. Sastoji se od redaka i stupaca: redci (n-torke) predstavljaju pojedinačne zapise, a stupci (atributi) njihova svojstva.

Tablica 1. Elementi relacijskog modela i njihov prikaz

Element relacijskog modela	Oblik prikaza
Relacija	Tablica
N-torka	Redak tablice
Atribut	Zaglavlje stupca tablice
Ključ	A set of attributes that uniquely identifies each row in a table, or is used to connect tables, or improves the efficiency of operations on tables
Domena	The set of all possible values an attribute may take
Shema relacije	The list of attribute names (table column headers) that define the structure of the relation

Razmotrimo, primjerice, tablicu sustava za praćenje stanja okoliša:

AirQuality(station_id, datetime, pollutant, value, unit).

Svaki redak tablice AirQuality predstavlja jedno mjerenje određenog onečišćivala, provedeno na određenoj postaji u određenom trenutku. Struktura tablice zahtijeva identifikator postaje, vremensku oznaku, naziv onečišćivala, brojčanu vrijednost i pripadnu mjernu jedinicu.

Strukturu tablice određuje shema. Ona definira atribute, njihove tipove podataka i pravila kojima moraju udovoljavati. Shema je nacrt baze podataka koji omogućuje smisleno tumačenje pohranjenih podataka. U širem značenju opisuje cijelu bazu, uključujući sve tablice, njihove veze i ograničenja. Upravo je ta izričito definirana struktura jedna od najvećih prednosti relacijskih baza: smanjuje dvosmislenost i omogućuje pouzdano postavljanje upita i analizu.

Primjerice, shema baze za praćenje okoliša može sadržavati:

- *Stations (station_id, name, region, latitude, longitude)*
- *AirQuality (station_id, datetime, pollutant, value, unit)*
- *Weather (station_id, datetime, temperature, humidity)*

Ključevi

Relacijski model oslanja se na ključeve radi jednoznačne identifikacije zapisa i održavanja logičkih veza među tablicama.

Primarni ključ (engl. primary key, PK) atribut je ili kombinacija atributa koja jednoznačno određuje svaki redak tablice. Dva retka ne mogu imati jednaku vrijednost primarnog ključa, a primarni ključ ne smije sadržavati NULL. Primjerice, u tablici Stations atribut station_id može biti primarni ključ. U složenijim tablicama, poput AirQuality, jedinstvenost se može postići složenim ključem (station_id, datetime, pollutant).

Strani ključ (engl. foreign key, FK) uspostavlja logičku vezu između dviju tablica. To je atribut jedne tablice koji upućuje na primarni ključ druge tablice i time osigurava dosljednost podataka. Tako svaka vrijednost station_id u tablici AirQuality mora odgovarati postojećoj vrijednosti station_id u tablici Stations. Bez stranih ključeva nastajale bi nevaljane reference, primjerice mjerenja za nepostojeće ili izbrisane postaje.

Kandidatski ključevi (engl. candidate keys) atributi su koji mogu jednoznačno identificirati retke pa mogu poslužiti kao primarni ključ. Tablica može imati više kandidatskih ključeva, no samo se jedan odabire kao primarni. Složeni ključevi (engl. composite keys) sastoje se od više atributa i koriste se kada nijedan pojedinačni atribut ne jamči jedinstvenost.

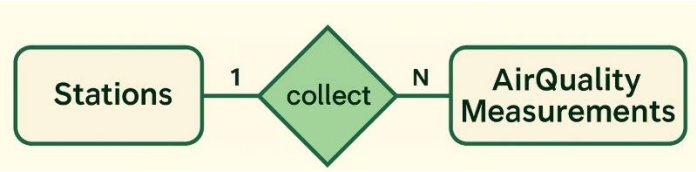
Ključevi omogućuju učinkovitost, pouzdanost i smisleno tumačenje podataka. Podupiru indeksiranje, sprječavaju dupliciranje i povezuju zapise iz različitih tablica, što je osobito važno u velikim bazama podataka o okolišu.

Veze

Veze opisuju kako su podaci jedne tablice povezani s podacima druge tablice. U relacijskom modelu razlikujemo više vrsta binarnih veza koje odražavaju odnose u stvarnom svijetu.

- **Jedan prema jedan (1:1)**
  U vezi jedan prema jedan svaki je zapis jedne tablice povezan s najviše jednim zapisom druge tablice. Takva je veza razmjerno rijetka, ali korisna kada neobavezne, osjetljive ili rijetko korištene podatke želimo izdvojiti u zasebnu tablicu.
- **Jedan prema više (1:N)**
  Veza jedan prema više najčešća je u relacijskim bazama podataka. Jedan redak prve tablice može biti povezan s više redaka druge tablice.

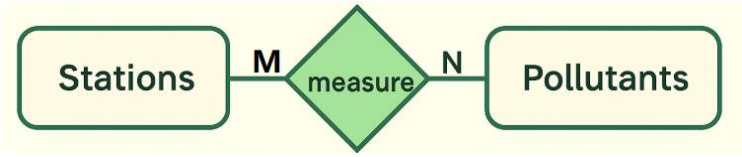
Primjerice, jedna mjerna postaja može svakodnevno prikupiti stotine mjerenja kvalitete zraka. U tom slučaju:
Stations nalazi se na strani „jedan", a AirQualityMeasurments na strani „više".



Slika 1. Primjer ER dijagrama u Chenovoj notaciji za vezu 1:N

Takve veze smanjuju redundanciju: metapodaci o postaji pohranjuju se samo jednom, a na njih se može višestruko upućivati.

- **Više prema više (M:N)**
  U vezi više prema više više redaka jedne tablice može odgovarati većem broju redaka druge tablice. Takva se veza u relacijskoj bazi ostvaruje preko povezne tablice koja pohranjuje parove stranih ključeva.
  Primjerice, jedna postaja može mjeriti više onečišćivala, a isto onečišćivalo može se mjeriti na više postaja. Riječ je o vezi M:N (slika 2.), koju možemo opisati poveznom tablicom StationPollutant(station_id, pollutant).



Slika 2. Primjer ER dijagrama u Chenovoj notaciji za vezu M:N

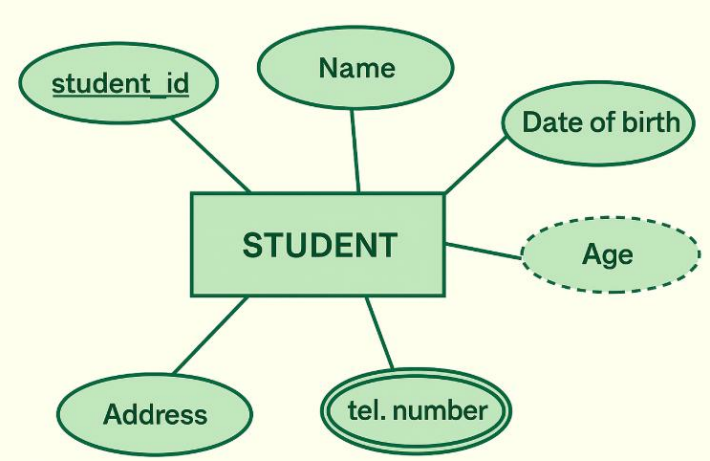
Razumijevanje veza ključno je za oblikovanje normaliziranih shema i pisanje ispravnih SQL upita, osobito pri spajanju tablica.

Kardinalnost veze određuje koliko se primjeraka jednog entiteta može povezati s pojedinim primjerkom drugog entiteta. U Chenovoj notaciji označava se brojevima uz entitete te pokazuje dopušteni broj sudionika u vezi.

Atributi

Atribut predstavlja svojstvo ili obilježje entiteta. Njegove vrijednosti pripadaju unaprijed definiranom skupu mogućih vrijednosti, koji nazivamo domenom atributa.

Pri konceptualnom oblikovanju baze razlikujemo vrste atributa prema strukturi, značenju i ponašanju. Glavne vrste prikazane su na slici 3. i opisane u tablici 2.



Slika 3. Primjer ER dijagrama entiteta Student u Chenovoj notaciji

Tablica 2. Vrste atributa

Vrsta atributa	Opis	Primjeri
Jednostavni (atomarni) atribut	An attribute that cannot be subdivided into smaller meaningful components.	*student_id, Last name*
Složeni atribut	An attribute that can be broken down into smaller sub-attributes, each with its own meaning.	*Address → street, building, apartment;* *Full name → last name, first name*
Jednovrijednosni atribut	An attribute that holds only one value for each entity instance.	*student_id, Passport number*
Viševrijednosni atribut	An attribute that may contain multiple values for the same entity instance.	*Phone numbers (mobile, home), Email addresses*

Izvedeni atribut	An attribute whose value is calculated from other stored data and is not kept directly in the database.	*Age (from Date of Birth), Number of credits (from list of courses)*
Ključni atribut	An attribute or group of attributes that uniquely identifies an entity instance.	*Primary Key: Student ID; Candidate Keys: Email, Record book number*

Ograničenja integriteta

Ograničenja integriteta pravila su koja baza podataka automatski provodi radi očuvanja točnosti, dosljednosti i pouzdanosti. Ona sprječavaju unos neispravnih podataka i osiguravaju valjanost veza među tablicama. Uobičajene vrste ograničenja jesu:

- **NOT NULL**
  Ograničenje NOT NULL zahtijeva da stupac uvijek sadržava vrijednost. Posebno je važno za atribute potrebne za analizu ili identifikaciju. Primjerice, vremenska oznaka mjerenja okoliša ne smije biti NULL.
- **UNIQUE**
  Ograničenje UNIQUE osigurava da dva retka nemaju jednaku vrijednost određenog atributa. Koristi se, primjerice, za serijske brojeve, korisnička imena ili oznake onečišćivala.
- **CHECK**
  Ograničenje CHECK zahtijeva da vrijednosti atributa zadovolje zadani logički uvjet. Primjerice, CHECK (value ≥ 0) sprječava pohranu negativnih koncentracija onečišćivala, koje u ovom kontekstu nisu fizikalno smislene.
- **Referencijalni integritet (putem stranih ključeva)**
  Referencijalni integritet zahtijeva da svaka vrijednost stranog ključa odgovara postojećem ključu referencirane tablice. Također sprječava brisanje referenciranih redaka ako se povezani zapisi prethodno ne obrade na odgovarajući način. To je osobito važno u složenim bazama s mnogo međusobno povezanih tablica.

Faze oblikovanja baze podataka

Oblikovanje baze podataka strukturiran je višefazni postupak kojim se potrebe za informacijama iz stvarnog svijeta pretvaraju u učinkovit i pouzdan sustav upravljanja podacima. Klasičan pristup obuhvaća četiri temeljne faze:

1. Sustavna analiza domene primjene
2. Konceptualno oblikovanje
3. Logičko oblikovanje
4. Fizičko oblikovanje

Svaka faza obuhvaća drukčiju razinu apstrakcije i pridonosi izgradnji robusnog, skalabilnog i dobro strukturiranog sustava baze podataka.

Oblikovanje počinje sustavnom analizom domene primjene, temeljem svih daljnjih faza. Projektant proučava poslovne procese, znanstveno područje i organizacijski kontekst kako bi utvrdio koje informacije sustav mora pohranjivati i kako će se one koristiti. Utvrđuje i dionike, njihove ciljeve, potrebe i očekivanja. Analiziraju se postojeći dokumenti, izvješća, obrasci, radni postupci i alati te način nastanka, obrade, prijenosa i uporabe podataka. Rezultat je strukturiran opis domene, njezinih ključnih objekata, svojstava, veza i pravila.

Nakon analize slijedi konceptualno oblikovanje, u kojem se struktura podataka prikazuje na visokoj razini apstrakcije, neovisno o odabranoj tehnologiji. Cilj je izraditi konceptualni podatkovni model, najčešće ER dijagramom ili UML dijagramom klasa. Projektant prepoznaje entitete i njihove atribute te određuje veze, kardinalnosti i obaveznost sudjelovanja. U model se ugrađuju i poslovna pravila utvrđena analizom. U ovoj fazi odgovara se na pitanje koje informacije baza treba predstavljati, a ne kako će se fizički pohranjivati.

Logičko oblikovanje pretvara konceptualni model u strukturu pogodnu za odabranu vrstu sustava za upravljanje bazom podataka (SUBP), najčešće relacijsku. Definiraju se tablice, primarni, kandidatski i strani ključevi, atributi i njihovi tipovi. Normalizacijom se smanjuje redundancija i izbjegavaju anomalije pri ažuriranju, umetanju i brisanju. Ograničenja integriteta osiguravaju ispravnost podataka. Logičko oblikovanje određuje kako će se konceptualni prikaz ostvariti u podatkovnom modelu odabranog SUBP-a.

Fizičko oblikovanje završna je faza u kojoj se logički model prilagođava konkretnom sklopovskom i programskom okruženju. Naglasak je na performansama, učinkovitosti i upravljanju podacima: biraju se strukture pohrane, organizacija datoteka, indeksi, particioniranje i grupiranje.

Planiraju se raspodjela memorije, optimizacija upita, sigurnosne kopije, arhiviranje i oporavak. Po potrebi se uvode kontrole pristupa, šifriranje i revizijski zapisi. Time se osigurava učinkovita i pouzdana fizička pohrana podataka.

Tablica 3. Sažetak faza oblikovanja baze podataka

Faza	Ključno pitanje	Rezultat
Sustavna analiza	*What is the domain? What do users need?*	Zahtjevi, poslovna pravila i razumijevanje domene
Konceptualno oblikovanje	*An attribute that can be broken down into smaller sub-attributes, each with its own meaning.*	ER dijagram ili konceptualna shema
Logičko oblikovanje	*How to represent this data in a DBMS data model?*	Relacijska shema: tablice, ključevi i ograničenja
Fizičko oblikovanje	*How to store and access the data efficiently?*	Indeksi, particije i konfiguracija pohrane

Normalizacija

Normalizacija je sustavan postupak oblikovanja relacijskih baza podataka kojim se podaci raspoređuju u dobro strukturirane tablice radi smanjenja redundancije i sprječavanja nedosljednosti. Svaka se činjenica nastoji pohraniti na odgovarajuće mjesto, a atributi trebaju ovisiti o ključu, cijelom ključu i ni o čemu osim o ključu. Time se smanjuje nepotrebna pohrana i sprječavaju anomalije pri ažuriranju, umetanju i brisanju.

Iako proizlazi iz teorije skupova i relacijske algebre, normalizacija je iznimno važna u praktičnom upravljanju podacima o okolišu. Skupovi podataka o kvaliteti zraka, meteorološkim mjerenjima, vrstama onečišćivala i mjernim postajama često su veliki, ponavljajući i međusobno povezani. Bez normalizacije višestruko pohranjeni nazivi postaja, definicije onečišćivala i mjerne jedinice lako postaju nedosljedni.

Normalizacija se provodi postupno, kroz niz normalnih formi s različitim zahtjevima za organizaciju podataka. U praksi su posebno važne prva, druga i treća normalna forma.

Prva normalna forma (1NF)

Prva normalna forma zahtijeva atomarnost podataka: svaka ćelija tablice sadržava jednu nedjeljivu vrijednost. Nisu dopuštene ponavljajuće skupine, ugniježđene tablice ni više vrijednosti u istom polju.

Primjerice, loše oblikovana tablica mjerenja može u jednom stupcu sadržavati više onečišćivala:

station_id	pollutants_measured
101	NO2, PM2.5, O3

Takav prikaz znatno otežava pretraživanje, filtriranje i indeksiranje. Za usklađenost s 1NF svako se onečišćivalo zapisuje u zaseban redak ili povezanu tablicu. Nakon normalizacije struktura postaje pregledna, dosljedna i pogodna za upite.

station_id	pollutants_measured
101	NO2
101	PM2.5
101	O3

Osiguravanjem atomarnosti 1NF stvara temelj za više normalne forme.

Druga normalna forma (2NF)

Druga normalna forma odnosi se na tablice sa složenim primarnim ključem. Zahtijeva da svaki neključni atribut ovisi o cijelom primarnom ključu, a ne samo o njegovu dijelu. Drugim riječima, djelomične funkcionalne ovisnosti nisu dopuštene.

Razmotrimo tablicu mjerenja kvalitete zraka s ključem (station_id, datetime):

Primjenjuje se na tablice sa složenim primarnim ključevima. Svaki neključni atribut mora ovisiti o cijelom ključu, a ne samo o njegovu dijelu; djelomične ovisnosti nisu dopuštene.

Razmotrimo tablicu mjerenja kvalitete zraka s ključem (station_id, datetime):

station_id	**datetime**	**pollutant**	**value**	**station_name**
101	2025-01-15 09:00	NO2	40	Zagreb Center

Druga normalna forma zahtijeva premještanje takvih atributa u zasebnu tablicu, u ovom slučaju Stations. Nakon normalizacije dobivamo:

AirQualityMeasurements(station_id, datetime, pollutant, value)

Stations(station_id, station_name)

Time uklanjamo redundanciju i osiguravamo da se svaki atribut nalazi na odgovarajućem mjestu.

Treća normalna forma (3NF)

Treća normalna forma uklanja tranzitivne ovisnosti: neključni atribut ne smije ovisiti o drugom neključnom atributu, nego samo o ključu.

Za prikaz treće normalne forme razmotrimo tablicu s podacima o meteorološkim postajama i regijama kojima pripadaju:

station_id	**station_name**	**region_name**	**region_manager**
201	Riverbank West	Central Basin	Dr. Alicia Moreno
202	Hilltop South	Mountain Highlands	Eng. Robert Davies

Primarni ključ je station_id. Atributi station_name, region_name i region_manager nisu ključni atributi. Međutim, region_manager ne ovisi izravno o station_id, nego o region_name, čime nastaje tranzitivna ovisnost:

station_id → region_name

region_name → region_manager.

Time se krši treća normalna forma.

Za uklanjanje tranzitivne ovisnosti tablicu dijelimo na dvije povezane tablice:

station_id (PK)	**station_name**	**region_name**
201	Riverbank West	Central Basin
202	Hilltop South	Mountain Highlands

region_name (PK)	region_manager
Central Basin	Dr. Alicia Moreno
Mountain Highlands	Eng. Robert Davies

Baza podataka sada zadovoljava treću normalnu formu:

- *station_id jednoznačno identificira svaku postaju*
- *region_name jednoznačno identificira svaku regiju*
- *region_manager ovisi samo o ključu region_name*
- nijedan neključni atribut ne ovisi o drugom neključnom atributu

Normalizacija čuva integritet i pouzdanost baze sprječavanjem logičkih anomalija koje nastaju pri nestrukturiranoj ili redundantnoj pohrani. Ista činjenica može biti ponovljena u više redaka ili se različite vrste podataka mogu neprikladno nalaziti u jednoj tablici. U informacijskim sustavima okoliša takve nedosljednosti mogu ozbiljno narušiti točnost analiza i izvješća.

Anomalija ažuriranja nastaje kada se ista informacija mora izmijeniti na više mjesta. Ako su naziv mjerne postaje ili regije ponovljeni u brojnim redcima, svaka se pojava mora dosljedno ažurirati. Propuštena izmjena uzrokuje proturječne zapise, koji se zatim prenose u izvješća, nadzorne ploče i modele.

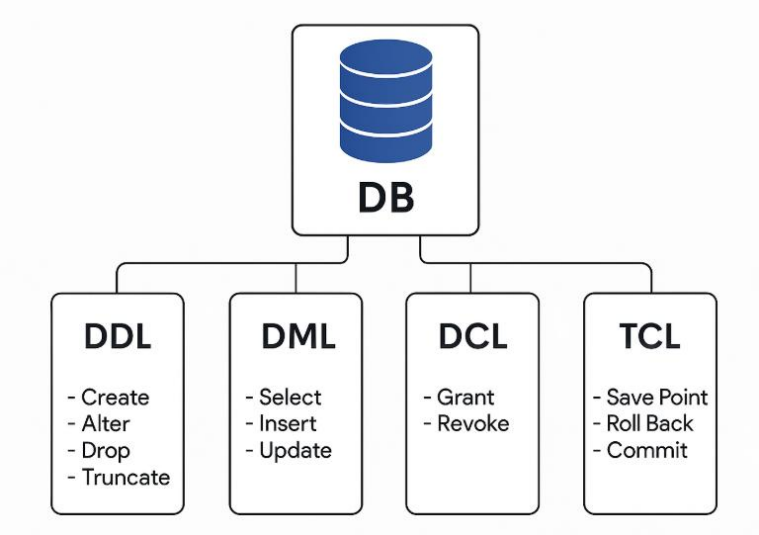
Anomalija umetanja pojavljuje se kada novu informaciju nije moguće unijeti bez drugih, nepovezanih podataka. Primjerice, novi se tip onečišćivala ili regija možda ne mogu dodati dok za njih ne postoji barem jedno mjerenje. Time se neprirodno ograničava redoslijed unosa podataka.

Anomalija brisanja suprotan je problem: uklanjanjem posljednjeg operativnog zapisa mogu se nenamjerno izgubiti važni opisni podaci. Ako se podaci o postaji čuvaju zajedno s mjerenjima, brisanjem posljednjeg mjerenja može nestati i opis postaje, iako ona i dalje postoji.

Raspoređivanjem podataka u normalizirane tablice izbjegavaju se takve anomalije. Svaka je činjenica pohranjena u odgovarajućoj tablici, a ostale tablice na nju upućuju ključevima. Baza time postaje dosljednija i lakša za održavanje te otpornija na pogrešna ažuriranja i brisanja. U primjenama poput dugoročnog praćenja kvalitete zraka, procjene slivova ili ekološkog modeliranja to je temelj pouzdanih, sljedivih i znanstveno valjanih podataka.

Strukturirani upitni jezik (SQL)

SQL (Structured Query Language) standardni je jezik za rad s relacijskim sustavima za upravljanje bazama podataka. Njime se definira struktura podataka, dohvaćaju i mijenjaju zapisi, uređuju prava pristupa i upravlja transakcijama. Iako je najpoznatiji po dohvaćanju podataka, SQL obuhvaća više skupina naredbi, koje se tradicionalno dijele na DDL, DML, DCL i TCL.



Slika 4. Struktura jezika SQL

1. Jezik za definiranje podataka (DDL)

DDL (Data Definition Language) služi za definiranje i mijenjanje strukture objekata baze podataka. Njegovim se naredbama opisuje shema baze: koje tablice postoje, koje atribute sadržavaju i kako su međusobno povezane.

- **CREATE služi za izradu novog objekta baze podataka, primjerice tablice, pogleda, indeksa ili cijele baze. Koristi se pri definiranju tablica s mjerenjima okoliša i pogleda koji sažimaju trendove onečišćenja.**

```
CREATE TABLE Stations (
    station_id INT PRIMARY KEY,
    name VARCHAR(100),
    region VARCHAR(50)
```

```
);
```

- **ALTER mijenja postojeći objekt baze podataka. Njime se može dodati novi stupac, promijeniti tip podataka radi veće preciznosti mjerenja ili prilagoditi ograničenja novim pravilima.**

```
ALTER TABLE AirQuality ADD COLUMN unit VARCHAR(20);
```

- **DROP potpuno uklanja objekt baze podataka. Kada tablica, indeks ili pogled više nisu potrebni, naredba ih briše iz sheme.**

```
DROP TABLE TempData;
```

Naredbe DDL-a čine strukturni okvir baze podataka i određuju način na koji će podaci biti pohranjeni i dostupni.

2. Jezik za manipulaciju podacima (DML)

DML (Data Manipulation Language) služi za svakodnevni rad sa zapisima: umetanje novih, dohvaćanje postojećih, izmjenu i brisanje zastarjelih ili pogrešnih podataka.

- **INSERT** adds new records to a table. As new sensor readings arrive from air quality stations or new metadata is entered for additional monitoring sites, INSERT statements populate the database.

INSERT dodaje nove retke u tablicu. Njome se unose nova očitanja senzora i metapodaci o dodatnim mjernim postajama.

```
INSERT INTO Stations VALUES (101, 'Zagreb-Center', 'Zagreb');
```

- **UPDATE** modifies existing records. If a station's metadata changes-such as an updated altitude, corrected coordinates, or a change in operator-UPDATE ensures that the stored information remains accurate.

  UPDATE mijenja postojeće zapise. Primjerice, promjenom nadmorske visine, ispravkom koordinata ili podatka o upravitelju postaje ažuriraju se njezini metapodaci.

```
UPDATE AirQuality
SET value = NULL
WHERE value < 0;
```

- **DELETE uklanja zapise iz tablice. Koristi se za brisanje pogrešnih ili zastarjelih podataka.**

```
DELETE FROM Stations WHERE region = 'Unknown';
```

SELECT dohvaća podatke iz jedne ili više tablica te omogućuje filtriranje, sortiranje, grupiranje i agregiranje. Analitičari ga koriste za otkrivanje prekoračenja graničnih vrijednosti i usporedbu onečišćivala među postajama.

- **SELECT** retrieves data from one or more tables, often applying filters, sorting, grouping, and aggregation. Environmental analysts might use SELECT to extract measurements that exceed regulatory thresholds or to compare pollutant levels across monitoring stations.

```
SELECT station_id, pollutant, value
FROM AirQuality
WHERE pollutant = 'PM2.5';
```

  - ***Sortiranje***

```
ORDER BY value DESC;
```

  - ***Filtriranje logičkim operatorima***

```
WHERE value > 50 AND pollutant = 'NO2';
```

  - ***Agregiranje i GROUP BY (za izvješćivanje o okolišu)***

```
SELECT station_id, pollutant, AVG(value) AS avg_value
FROM AirQuality
GROUP BY station_id, pollutant;
```

  - ***HAVING – filtriranje agregiranih podataka***

```
HAVING AVG(value) > 40;
```

  - ***Unutarnje spajanje (INNER JOIN)***

```
SELECT s.name, a.pollutant, a.value
FROM AirQuality a
JOIN Stations s ON a.station_id = s.station_id;
```

- ***Lijevo vanjsko spajanje (LEFT JOIN)***

  Primjer: želimo prikazati sve mjerne postaje, uključujući one za koje još nema zabilježenih mjerenja. Tablica 1: Stations(station_id, station_name). Tablica 2: AirQualityMeasurements(station_id, pollutant, value).

```
SELECT s.station_id, s.station_name, m.pollutant, m.value
FROM Stations AS s
LEFT JOIN AirQualityMeasurements AS m
ON s.station_id = m.station_id;
```

- ***Desno vanjsko spajanje (RIGHT JOIN)***

  Primjer: želimo prikazati sva onečišćivala, uključujući ona koja još nisu dodijeljena nijednoj postaji. Tablica 1: AirQualityMeasurements(station_id, pollutant, value). Tablica 2: Pollutants(pollutant, description).

```
SELECT m.station_id, p.pollutant, p.description, m.value
FROM AirQualityMeasurements AS m
RIGHT JOIN Pollutants AS p
ON m.pollutant = p.pollutant;
```

- **Podupiti**

  Ugniježđene analize, primjerice prekoračenja graničnih vrijednosti EU-a:

```
SELECT *
FROM AirQuality
WHERE value > (
      SELECT threshold
      FROM EU_Standards
      WHERE pollutant = AirQuality.pollutant
);
```

DML je operativni dio SQL-a koji omogućuje neposredan rad s pohranjenim podacima.

3. Jezik za upravljanje pristupom podacima (DCL)

DCL (Data Control Language) regulira prava pristupa objektima baze podataka. U višekorisničkim okruženjima, osobito znanstvenim i javnim sustavima u kojima se podaci o okolišu moraju dijeliti, ali i štititi, DCL osigurava da pojedine radnje mogu obavljati samo ovlašteni korisnici.

- **GRANT dodjeljuje prava korisnicima ili ulogama za izvođenje pojedinih operacija nad objektima. Istraživačima se, primjerice, može dopustiti čitanje mjerenja bez prava izmjene metapodataka postaja.**

```
GRANT SELECT
ON AirQualityMeasurements
TO researcher_1;
GRANT SELECT, INSERT, UPDATE, DELETE
ON Stations
TO env_admin_role;
```

- **REVOKE oduzima prethodno dodijeljena prava. Važan je pri promjeni odgovornosti ili ograničavanju pristupa radi sigurnosti i usklađenosti s pravilima.**

```
REVOKE INSERT
ON AirQualityMeasurements
FROM researcher_1;
```

DENY izričito zabranjuje određene radnje, čak i ako ih druga dodijeljena prava dopuštaju. Time se provode stroža sigurnosna pravila.

- **DENY** specifically forbids certain actions, overriding any permissions that might otherwise allow them. It is a strict measure that enforces security policies even if multiple permission layers are in place.

```
DENY DELETE
ON Pollutants
TO researcher_2;
```

Naredbe DCL-a podupiru povjerljivost i integritet podataka te kontroliranu suradnju među timovima.

4. Jezik za upravljanje transakcijama (TCL)

TCL (Transaction Control Language) omogućuje da se skupine povezanih DML operacija izvršavaju atomarno: ili uspijevaju u cijelosti ili se poništavaju bez djelomičnih promjena. To je osobito važno u povezanim skupovima podataka o okolišu, gdje bi nedosljednosti mogle narušiti rezultate analize.

- **COMMIT potvrđuje transakciju i trajno primjenjuje sve promjene od prethodne potvrde. Nakon toga umetnuta mjerenja, izmijenjeni metapodaci i izbrisani zapisi čine službeno stanje baze.**

```
BEGIN TRANSACTION;

INSERT INTO AirQualityMeasurements (station_id, pollutant, value)
VALUES (101, 'PM2.5', 23.5);

COMMIT;
```

- **ROLLBACK poništava sve promjene trenutačne transakcije. Ako se tijekom ažuriranja povezanih tablica pojavi pogreška, baza se vraća u prethodno stabilno stanje.**

```
BEGIN TRANSACTION;
UPDATE AirQualityMeasurements
SET value = -10      -- invalid value
WHERE measurement_id = 455;
-- Error detected → revert everything
ROLLBACK;
```

- **SAVEPOINT stvara imenovanu točku spremanja unutar transakcije i omogućuje djelomično poništavanje. To je korisno pri uvozu velikih skupina senzorskih podataka kada se treba vratiti na međukorak, a ne ponavljati cijeli postupak.**

```
BEGIN TRANSACTION;
UPDATE Stations
SET region = 'Central Basin'
WHERE station_id = 101;
SAVEPOINT sp1;

UPDATE Stations
SET region = 'Mountain Highlands'
WHERE station_id = 102;

-- Decide to undo only the second update
ROLLBACK TO sp1;

COMMIT;
```

TCL omogućuje dosljedan i pouzdan rad s podacima, čak i pri složenim višekoračnim operacijama.

ACID svojstva transakcija

Podaci o kvaliteti zraka, meteorološka opažanja, senzorski zapisi i regionalni metapodaci često su povezani i neprekidno se ažuriraju. Oštećenje ili djelomično ažuriranje i jednog dijela takvih podataka može ugroziti analize i dugoročno praćenje.

Kako bi se višekoračne operacije izvršavale predvidljivo i dosljedno, suvremeni sustavi koriste transakcije – skupine operacija koje se obrađuju kao jedna cjelina. Njihovu ispravnost opisuju četiri svojstva ACID: atomarnost (Atomicity), dosljednost (Consistency), izoliranost (Isolation) i trajnost (Durability).

1. Atomarnost

Atomarnost zahtijeva da se transakcija promatra kao nedjeljiva cjelina: ili uspiju sve njezine operacije ili se nijedna ne primijeni.

Transakcija može obuhvatiti umetanje mjerenja, ažuriranje kalibracijskih zapisa i bilježenje događaja. Atomarnost sprječava da se dovrši samo dio postupka. Ako nastane pogreška ili prekid sustava, cijela se transakcija poništava i vraća prethodno stabilno stanje.

Primjer: postaja šalje skupinu mjerenja kvalitete zraka. Ako unos makar jednoga mjerenja ne uspije zbog pogrešnog formata, nijedan se zapis ne unosi, čime se izbjegavaju djelomični i potencijalno zavaravajući podaci.

Atomarnost štiti složene postupke od djelomičnih ažuriranja koja bi mogla iskriviti analizu okoliša.

2. Dosljednost

Dosljednost osigurava da svaka transakcija prevodi bazu iz jednog valjanog stanja u drugo, uz očuvanje svih pravila, ograničenja i veza.

Ta pravila obuhvaćaju tipove podataka, primarne i strane ključeve, ograničenja CHECK i pravila znanstvene provjere. Valjana baza poštuje ih prije i nakon svake transakcije.

Primjer: ako koncentracije onečišćivala ne smiju biti negativne, pokušaj unosa vrijednosti -15 mora završiti neuspjehom. Baza odbija neispravan podatak radi očuvanja znanstvene točnosti.

Dosljednost jamči poštovanje fizikalnih, regulatornih i logičkih ograničenja.

3. Izoliranost

U stvarnim sustavima mnogi korisnici i procesi istodobno pristupaju bazi. Izoliranost osigurava da izvršavanje jedne transakcije ne ometa drugu i da drugi korisnici ne vide njezine međurezultate prije dovršetka.

Bez izoliranosti dvije transakcije koje istodobno mijenjaju metapodatke o postaji ili unose preklapajuće skupove mjerenja mogle bi jedna drugoj utjecati na rezultat.

Primjer: dva istraživača istodobno mijenjaju različite atribute tablice Stations. Izoliranost sprječava sukobe i neželjeno prepisivanje promjena.

Većina sustava podržava više razina izolacije, primjerice:

- Read Uncommitted
- Read Committed

- Repeatable Read
- Serializable

Više razine pružaju jača jamstva, ali mogu ograničiti istodobno izvršavanje. U sustavima koji zaprimaju velike količine senzorskih podataka važno je odabrati odgovarajuću ravnotežu.

4. Trajnost

Trajnost jamči da potvrđena transakcija ostaje trajno pohranjena, čak i u slučaju kvara sklopovlja, nestanka napajanja ili pada sustava.

Sustavi baza podataka ostvaruju trajnost primjenom:

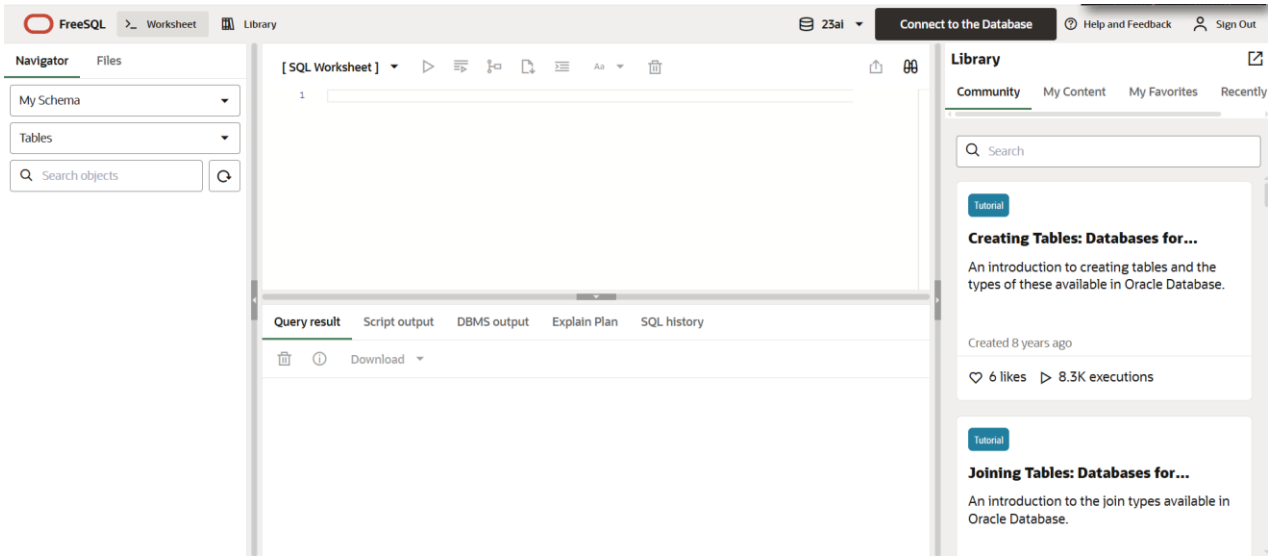
- dnevnika zapisivanja unaprijed (write-ahead log)
- kontrolnih točaka (checkpointing)
- redundantne pohrane
- mehanizama automatskog oporavka

Primjer: nakon unosa meteoroloških mjerenja za cijeli dan i naredbe COMMIT podaci moraju ostati sačuvani čak i ako odmah zatim nastane kvar. Trajnost omogućuje očuvanje podataka tijekom višegodišnjeg praćenja i izvješćivanja.

Svojstva ACID – atomarnost, dosljednost, izoliranost i trajnost – osiguravaju da su operacije nad podacima potpune, valjane, međusobno neometane i trajne.

Primjer uporabe

Za učenje SQL-a koristit ćemo mrežni alat Oracle FreeSQL (https://freesql.com/). On studentima omogućuje besplatan pristup stvarnoj bazi Oracle u oblaku te vježbanje SQL-a i relacijskih pojmova u profesionalnom okruženju. Za rad se potrebno prijaviti u postojeći račun ili otvoriti novi.https://freesql.com/



Slika 5. Sučelje Oracle FreeSQL

Prije praktičnog rada sa SQL-om potrebno je razumjeti kakvu bazu gradimo i zašto.

Tijekom nastave izradit ćemo oglednu relacijsku bazu podataka za praćenje kvalitete zraka u Varšavi u Poljskoj. Namijenjena je učenju i slijedi logiku stvarnih informacijskih sustava okoliša koje koriste javna tijela i istraživačke ustanove. Vrijednosti su pojednostavljene, ali struktura odražava stvarne veze i ograničenja.

Baza prikazuje kako mjerne postaje prikupljaju podatke o onečišćenju zraka, kako se postaje opisuju metapodacima te kako se definiraju i reguliraju vrste onečišćivala. Svaki se pojam pohranjuje u zasebnu tablicu prema načelima relacijskog modela.

Središnji dio baze čine mjerne postaje, odnosno lokacije u Varšavi na kojima su postavljeni senzori kvalitete zraka. Svaka postaja ima identifikator, naziv, zemljopisni položaj te dodatne podatke poput regije i upravitelja. Podaci o postajama čuvaju se neovisno o mjerenjima.

Druga važna sastavnica jest šifrarnik onečišćivala. Dušikov dioksid (NO_2), lebdeće čestice (PM10, PM2.5), ozon (O_3) i sumporov dioksid (SO_2) definiraju se samo jednom te se dosljedno koriste u cijeloj bazi. Time se smanjuje dupliciranje i olakšava usporedba mjerenja.

Mjerenja čine glavni činjenični dio baze. Svaki zapis sadržava koncentraciju određenog onečišćivala na određenoj postaji i u određenom trenutku. Povezanost ključevima omogućuje upite koji ujedinjuju lokaciju, vrijeme i vrstu onečišćivala.

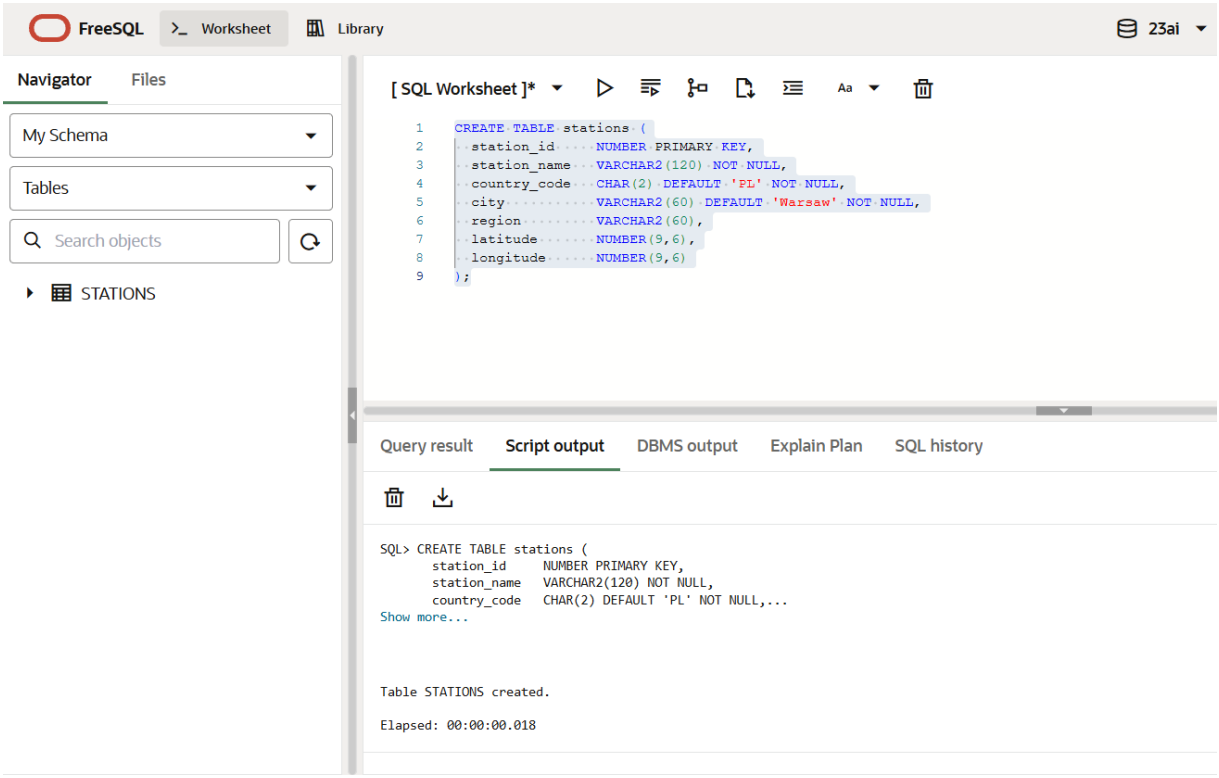
Baza sadržava i tablicu s pojednostavljenim europskim graničnim vrijednostima kvalitete zraka. Ona omogućuje otkrivanje prekoračenja i usporedbu izmjerenih koncentracija s propisanim vrijednostima.

Cijela struktura slijedi načela normalizacije i referencijalnog integriteta: svaki je pojam prikazan zasebnom tablicom, a veze su izričito definirane. Takva je baza dosljedna, proširiva i pogodna za analitičke SQL upite sa spajanjima, agregiranjem i podupitima.

1. SQL naredba CREATE

Izradimo bazu tako da postupno definiramo tablice. Izradite tablicu stations: upišite SQL kod u polje SQL Worksheet i pritisnite Run Statement.

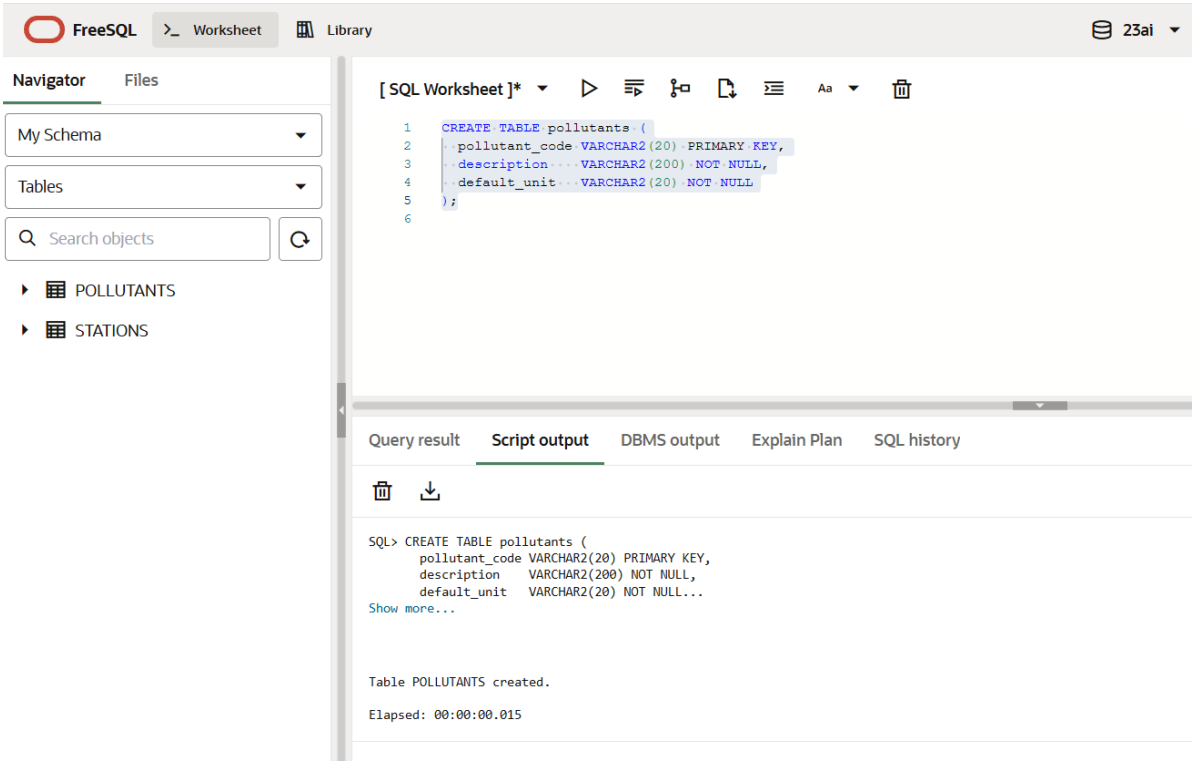
```
CREATE TABLE stations (
  station_id     NUMBER PRIMARY KEY,
  station_name   VARCHAR2(120) NOT NULL,
  country_code   CHAR(2) DEFAULT 'PL' NOT NULL,
  city           VARCHAR2(60) DEFAULT 'Warsaw' NOT NULL,
  region         VARCHAR2(60),
  latitude       NUMBER(9,6),
  longitude      NUMBER(9,6)
);
```



Slika 6. Izrada tablice stations

Zatim izradite tablicu pollutants

```
CREATE TABLE pollutants (
  pollutant_code VARCHAR2(20) PRIMARY KEY,
  description    VARCHAR2(200) NOT NULL,
  default_unit   VARCHAR2(20) NOT NULL
);
```



Slika 7. Izrada tablice pollutants

Izrada tablice mjerenja kvalitete zraka

```
CREATE TABLE air_quality_measurements (
  measurement_id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  station_id     NUMBER NOT NULL,
  pollutant_code VARCHAR2(20) NOT NULL,
  measurement_ts TIMESTAMP NOT NULL,
  value          NUMBER,
  unit           VARCHAR2(20),

  CONSTRAINT fk_station
    FOREIGN KEY (station_id)
    REFERENCES stations(station_id),
```

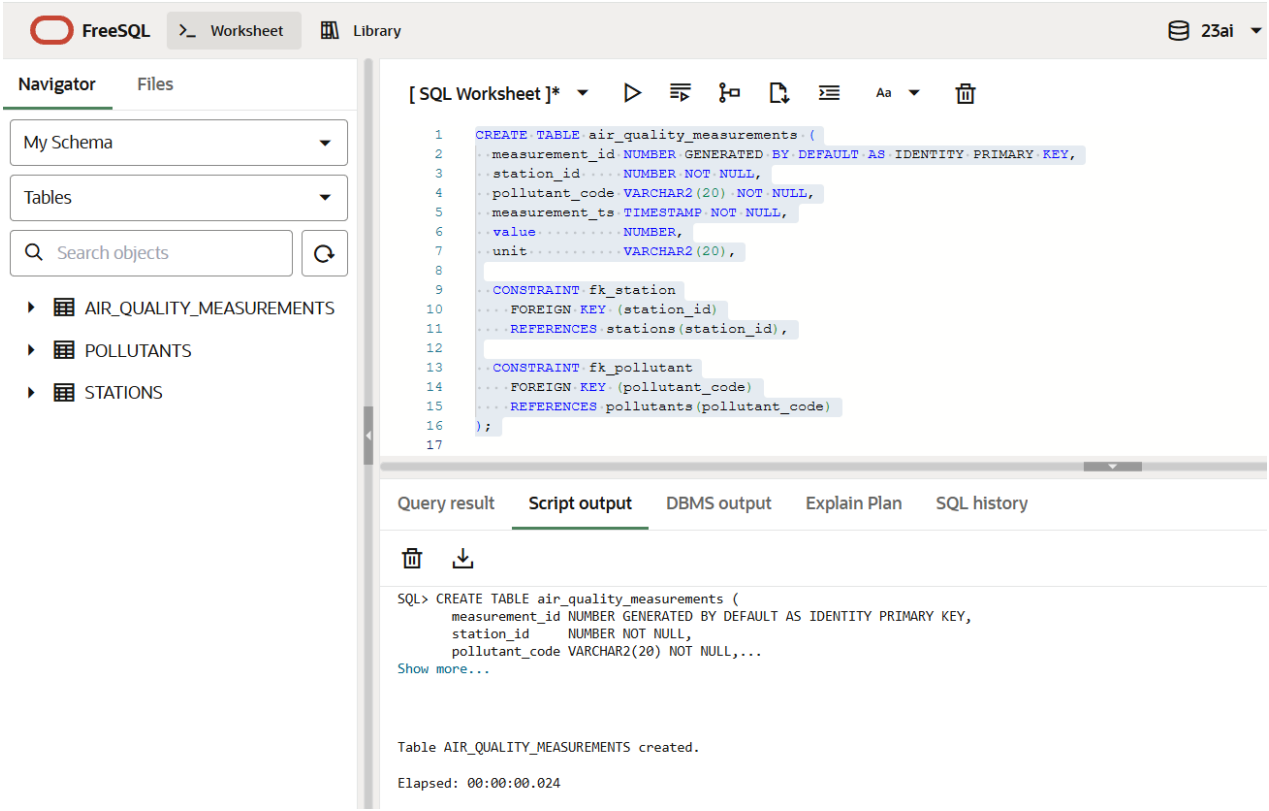
Slika 8. Izrada tablice mjerenja kvalitete zraka

```
  CONSTRAINT fk_pollutant
```

U ovoj su fazi tablice izrađene. Njihovu prisutnost provjerite u lijevom oknu Navigator, gdje trebaju biti vidljive STATIONS, POLLUTANTS i AIR_QUALITY_MEASUREMENTS.

REFERENCES pollutants(pollutant_code)

2. Operacije INSERT



2. SQL naredba INSERT

INSERT INTO stations VALUES

(101, 'Warsaw-Center', 'PL', 'Warsaw', 'Mazowieckie', 52.229676, 21.012229);

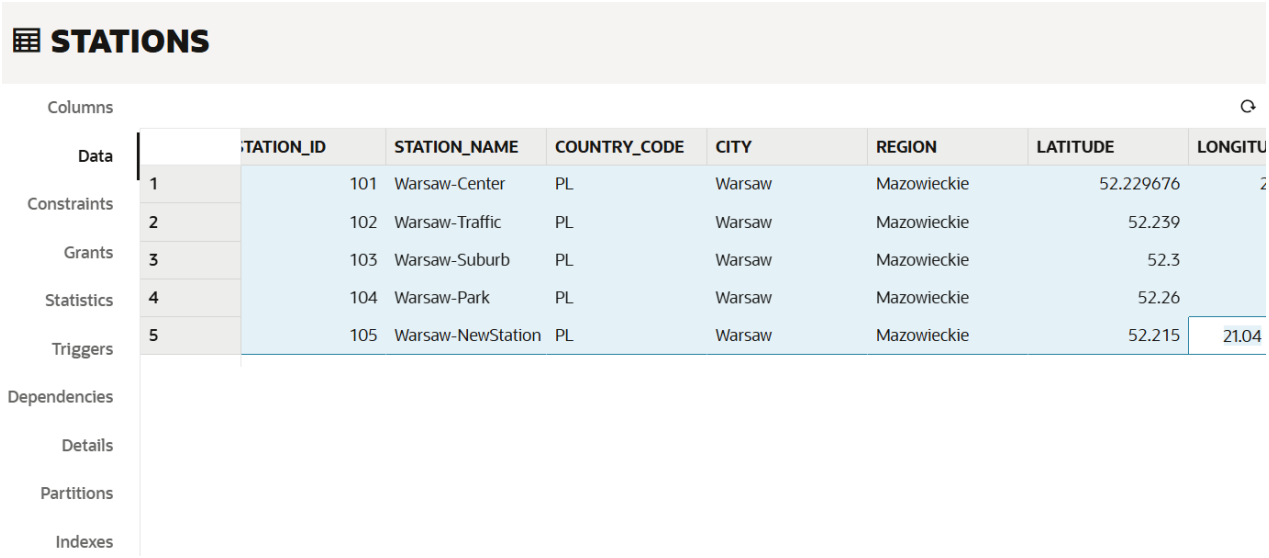
INSERT INTO stations VALUES

(102, 'Warsaw-Traffic', 'PL', 'Warsaw', 'Mazowieckie', 52.239000, 21.005000);

INSERT INTO stations VALUES

```
(103, 'Warsaw-Suburb', 'PL', 'Warsaw', 'Mazowieckie', 52.300000, 20.900000);

INSERT INTO stations VALUES

(104, 'Warsaw-Park', 'PL', 'Warsaw', 'Mazowieckie', 52.260000, 21.020000);

-- Station without measurements (for LEFT JOIN testing)

INSERT INTO stations VALUES

(105, 'Warsaw-NewStation', 'PL', 'Warsaw', 'Mazowieckie', 52.215000, 21.040000);
```

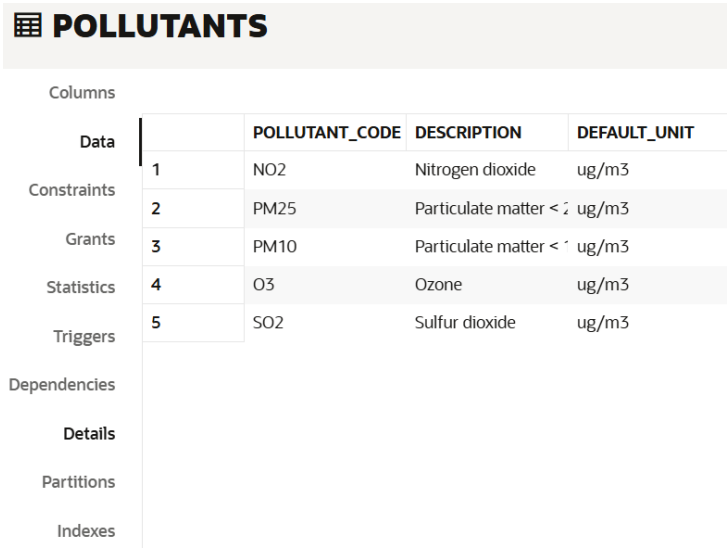
Nakon izvršavanja naredbi desnom tipkom miša kliknite tablicu STATIONS u lijevom oknu i otvorite je. Možete provjeriti jesu li podaci uspješno uneseni (slika 9.).



Slika 9. Rezultat unosa podataka u tablicu stations

```
INSERT INTO pollutants VALUES
('NO2', 'Nitrogen dioxide', 'ug/m3');

INSERT INTO pollutants VALUES
('PM25', 'Particulate matter < 2.5 µm', 'ug/m3');
```

```
INSERT INTO pollutants VALUES
('PM10', 'Particulate matter < 10 µm', 'ug/m3');
INSERT INTO pollutants VALUES
('O3', 'Ozone', 'ug/m3');

-- Pollutant without measurements (for testing RIGHT JOIN)
INSERT INTO pollutants VALUES
('SO2', 'Sulfur dioxide', 'ug/m3');
```



Slika 10. Rezultat unosa podataka o onečišćivalima

```
INSERT INTO air_quality_measurements
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(101, 'NO2', TIMESTAMP '2025-03-01 08:00:00', 45, 'ug/m3');

INSERT INTO air_quality_measurements
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(102, 'PM25', TIMESTAMP '2025-03-01 08:00:00', 28, 'ug/m3');

INSERT INTO air_quality_measurements
```

```
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(101, 'NO2', TIMESTAMP '2025-03-01 09:00:00', 78, 'ug/m3');

INSERT INTO air_quality_measurements
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(102, 'NO2', TIMESTAMP '2025-03-01 08:00:00', 210, 'ug/m3');

INSERT INTO air_quality_measurements
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(103, 'PM25', TIMESTAMP '2025-03-01 08:00:00', 18, 'ug/m3');

INSERT INTO air_quality_measurements
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(104, 'PM10', TIMESTAMP '2025-03-01 08:00:00', 62, 'ug/m3');

INSERT INTO air_quality_measurements
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(104, 'O3', TIMESTAMP '2025-03-01 13:00:00', 130, 'ug/m3');

-- Incorrect measurement (for UPDATE and DELETE testing)
INSERT INTO air_quality_measurements
(station_id, pollutant_code, measurement_ts, value, unit)
VALUES
(101, 'PM25', TIMESTAMP '2025-03-01 10:00:00', -7, 'ug/m3');
```

AIR_QUALITY_MEASUREMENTS

Columns | Data | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

	MEASUREMENT_ID	STATION_ID	POLLUTANT_CODE	MEASUREMENT_TS	VALUE	UNIT
1	1	101	NO2	2025-03-01T08:00:(	45	ug/m3
2	2	102	PM25	2025-03-01T08:00:(	28	ug/m3
3	3	101	NO2	2025-03-01T09:00:(	78	ug/m3
4	4	102	NO2	2025-03-01T08:00:(	210	ug/m3
5	5	103	PM25	2025-03-01T08:00:(	18	ug/m3
6	6	104	PM10	2025-03-01T08:00:(	62	ug/m3
7	7	104	O3	2025-03-01T13:00:(	130	ug/m3
8	8	101	PM25	2025-03-01T10:00:(	-7	ug/m3

Slika 11. Rezultat unosa podataka o mjerenjima

3. SQL naredba UPDATE i DELETE

Naredbom UPDATE možemo ispraviti neispravne vrijednosti mjerenja tako da negativne vrijednosti zamijenimo s NULL, a zatim naredbom DELETE uklonimo nevaljane zapise:

```
UPDATE air_quality_measurements
SET value = NULL
WHERE value < 0;
DELETE FROM air_quality_measurements
WHERE value IS NULL;
```

AIR_QUALITY_MEASUREMENTS

Columns | Data | Constraints | Grants | Statistics | Triggers | Dependencies | Details

	MEASUREMENT_ID	STATION_ID	POLLUTANT_CODE	MEASUREMENT_TS	VALUE	UNIT
1	1	101	NO2	2025-03-01T08:00:(	45	ug/m3
2	2	102	PM25	2025-03-01T08:00:(	28	ug/m3
3	3	101	NO2	2025-03-01T09:00:(	78	ug/m3
4	4	102	NO2	2025-03-01T08:00:(	210	ug/m3
5	5	103	PM25	2025-03-01T08:00:(	18	ug/m3
6	6	104	PM10	2025-03-01T08:00:(	62	ug/m3
7	7	104	O3	2025-03-01T13:00:(	130	ug/m3

Slika 12. Rezultat uklanjanja neispravnih vrijednosti mjerenja

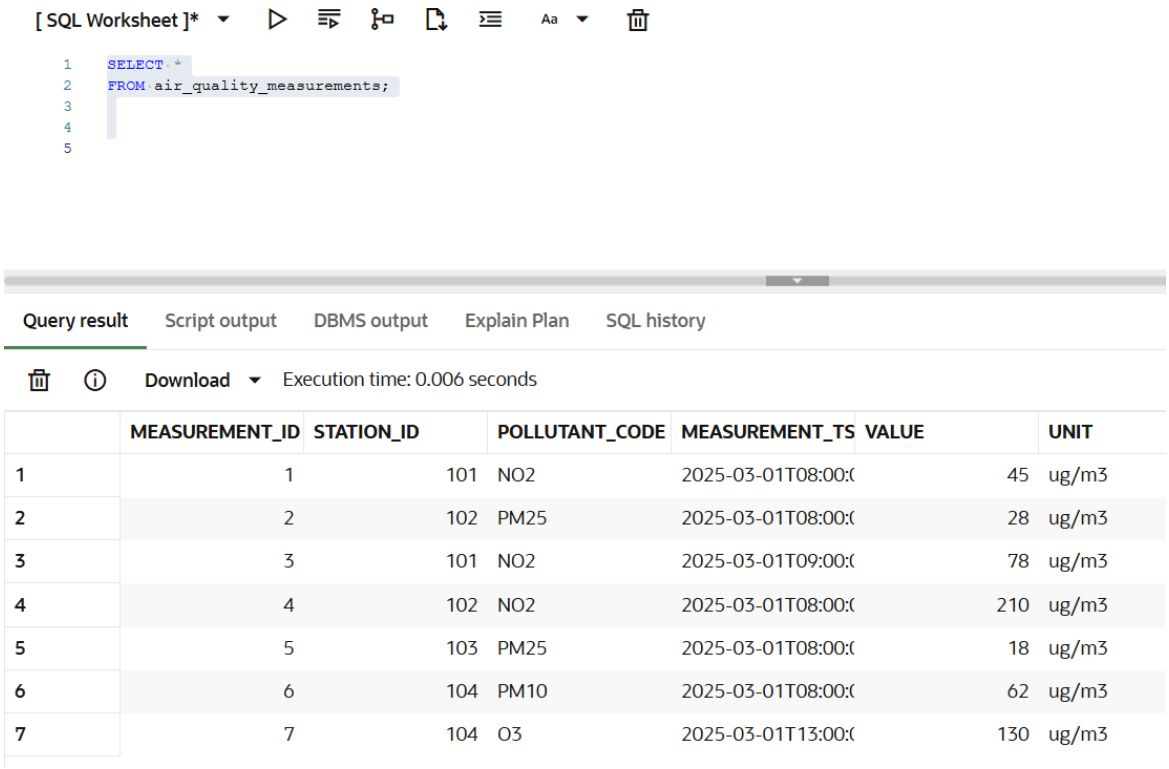
Napišite sličnu naredbu za brisanje postaja kojima nedostaje vrijednost regije (NULL).

4. SQL naredba SELECT

Upoznali smo naredbe DML-a, pa sada isprobajmo dohvat podataka naredbom SELECT.

Jednostavni SELECT:

```
SELECT *
FROM air_quality_measurements;
```



	MEASUREMENT_ID	STATION_ID	POLLUTANT_CODE	MEASUREMENT_TS	VALUE	UNIT
1	1	101	NO2	2025-03-01T08:00:(	45	ug/m3
2	2	102	PM25	2025-03-01T08:00:(	28	ug/m3
3	3	101	NO2	2025-03-01T09:00:(	78	ug/m3
4	4	102	NO2	2025-03-01T08:00:(	210	ug/m3
5	5	103	PM25	2025-03-01T08:00:(	18	ug/m3
6	6	104	PM10	2025-03-01T08:00:(	62	ug/m3
7	7	104	O3	2025-03-01T13:00:(	130	ug/m3

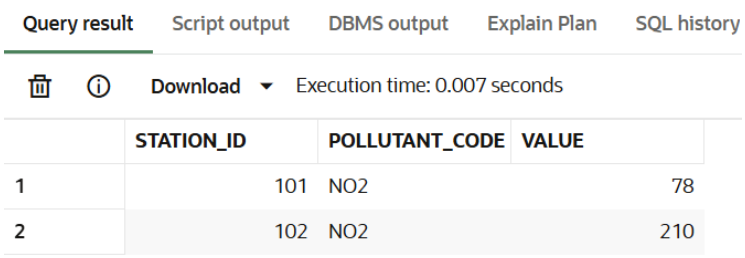
Slika 13. Rezultat izvršavanja upita

Provjerite kako se mijenja rezultat sljedećeg upita ako podatke sortiramo izrazom ORDER BY:

```
SELECT station_id, pollutant_code, value
FROM air_quality_measurements
ORDER BY value DESC;
```

Primijenimo filtriranje pomoću WHERE i logičkih operatora kako bismo dohvatili samo podatke o odabranom onečišćivalu:

```
SELECT station_id, pollutant_code, value
FROM air_quality_measurements
WHERE value > 50
  AND pollutant_code = 'NO2';
```



Slika 14. Rezultat izvršavanja upita

Primijenimo agregatne funkcije i grupiranje. Kako će se promijeniti rezultat?

```
SELECT station_id, pollutant_code, AVG(value) AS avg_value
FROM air_quality_measurements
WHERE value IS NOT NULL
GROUP BY station_id, pollutant_code;
```

Za prikaz onečišćivala s prosječnom vrijednošću većom od 40 upotrijebimo HAVING:

```
SELECT station_id, pollutant_code, AVG(value) AS avg_value
FROM air_quality_measurements
WHERE value IS NOT NULL
GROUP BY station_id, pollutant_code
HAVING AVG(value) > 40;
```

Query result | Script output | DBMS output | Explain Plan | SQL history

Download | Execution time: 0.073 seconds

	STATION_ID	POLLUTANT_CODE	AVG_VALUE
1	101	NO2	61.5
2	102	NO2	210
3	104	PM10	62
4	104	O3	130

Slika 15. Rezultat izvršavanja upita

5. SQL naredba JOIN

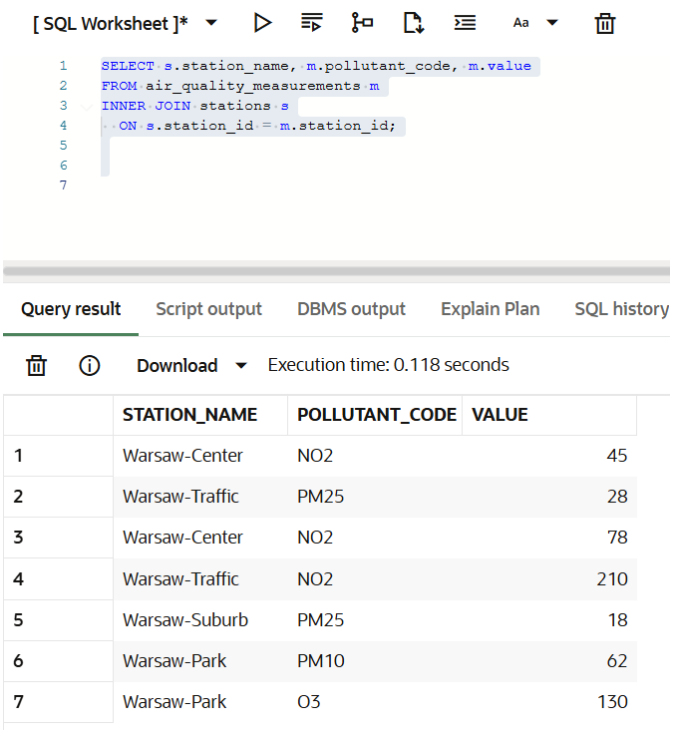
U relacijskoj bazi informacije su rijetko smještene u samo jednoj tablici. Podaci su raspoređeni u više tablica koje predstavljaju različite pojmove stvarnog svijeta. Za učinkovitu analizu često treba povezati retke različitih tablica u jedinstven rezultat, što se postiže operacijom JOIN.

JOIN određuje kako se redci dviju tablica povezuju na temelju uvjeta, najčešće primarnog ključa u jednoj i stranog ključa u drugoj tablici. U našoj oglednoj bazi mjerenja kvalitete zraka povezana su s postajama atributom station_id. Tako upit istodobno može dohvatiti vrijednosti mjerenja i opisne podatke postaja.

5.1 INNER JOIN vraća samo retke za koje u obje tablice postoji odgovarajuća vrijednost. Redci bez podudaranja izostavljaju se.

Pri spajanju mjerenja kvalitete zraka i mjernih postaja INNER JOIN vraća samo mjerenja povezana s poznatom postajom. To je najčešća vrsta spajanja, prikladna kada nepotpune ili nepovezane zapise treba izostaviti.

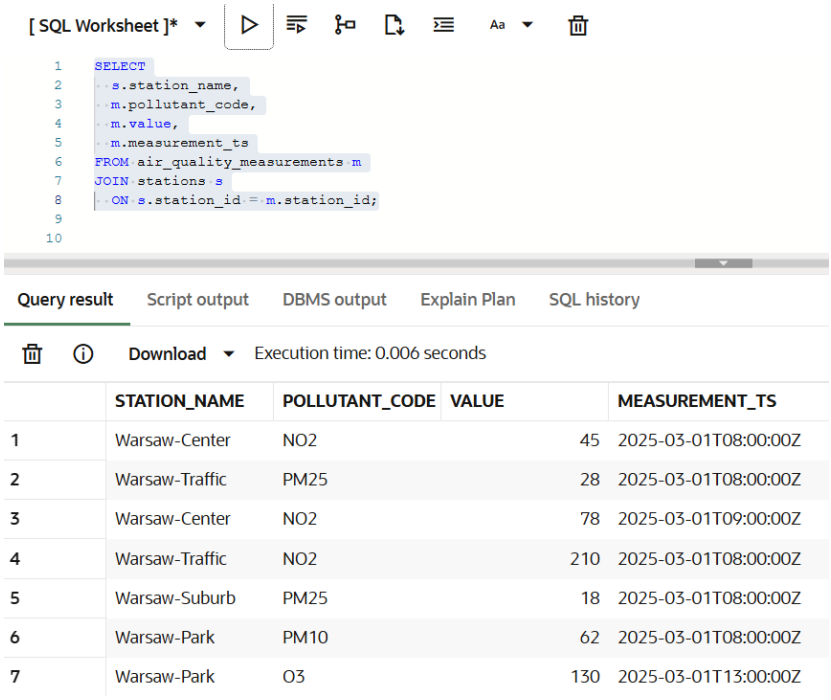
U pojmovnom smislu INNER JOIN odgovara na pitanje: „Koji se zapisi podudaraju u objema tablicama?”



	STATION_NAME	POLLUTANT_CODE	VALUE
1	Warsaw-Center	NO2	45
2	Warsaw-Traffic	PM25	28
3	Warsaw-Center	NO2	78
4	Warsaw-Traffic	NO2	210
5	Warsaw-Suburb	PM25	18
6	Warsaw-Park	PM10	62
7	Warsaw-Park	O3	130

Slika 16. Rezultat upita INNER JOIN

Ako se u upitu navodi samo JOIN, podrazumijeva se INNER JOIN (slika 17.).



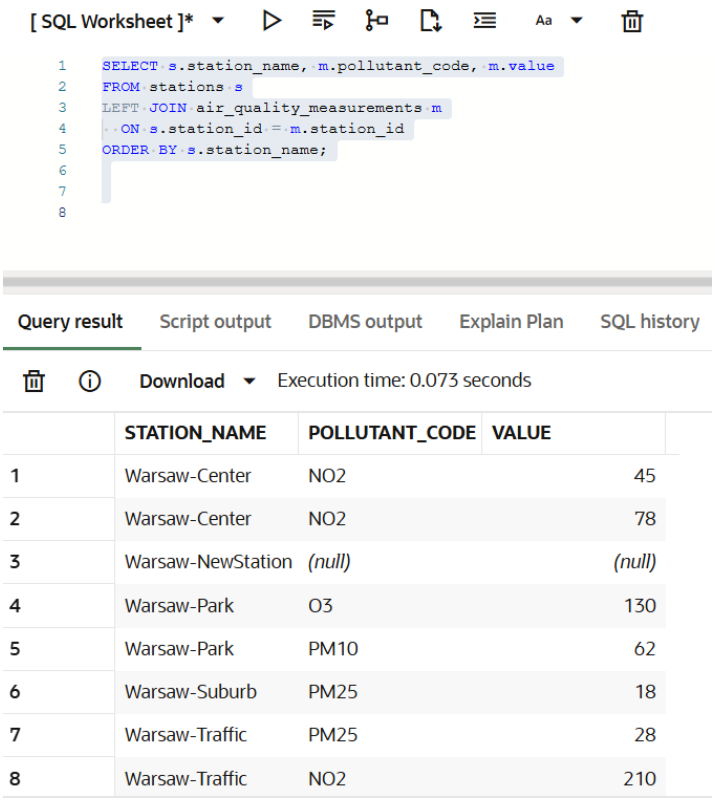
	STATION_NAME	POLLUTANT_CODE	VALUE	MEASUREMENT_TS
1	Warsaw-Center	NO2	45	2025-03-01T08:00:00Z
2	Warsaw-Traffic	PM25	28	2025-03-01T08:00:00Z
3	Warsaw-Center	NO2	78	2025-03-01T09:00:00Z
4	Warsaw-Traffic	NO2	210	2025-03-01T08:00:00Z
5	Warsaw-Suburb	PM25	18	2025-03-01T08:00:00Z
6	Warsaw-Park	PM10	62	2025-03-01T08:00:00Z
7	Warsaw-Park	O3	130	2025-03-01T13:00:00Z

Slika 17. Rezultat upita INNER JOIN

5.2 LEFT JOIN vraća sve retke lijeve tablice, čak i kada za njih ne postoji odgovarajući redak desne tablice. Ako podudaranja nema, stupci desne tablice sadržavaju NULL.

Takvo je spajanje korisno za prepoznavanje nepotpunih podataka. Primjerice, pri povezivanju postaja i mjerenja LEFT JOIN prikazuje sve postaje, uključujući one bez zabilježenih mjerenja.

LEFT JOIN odgovara na pitanje: „Koji zapisi postoje u lijevoj tablici i koji su im povezani podaci dostupni u desnoj?"



	STATION_NAME	POLLUTANT_CODE	VALUE
1	Warsaw-Center	NO2	45
2	Warsaw-Center	NO2	78
3	Warsaw-NewStation	*(null)*	*(null)*
4	Warsaw-Park	O3	130
5	Warsaw-Park	PM10	62
6	Warsaw-Suburb	PM25	18
7	Warsaw-Traffic	PM25	28
8	Warsaw-Traffic	NO2	210

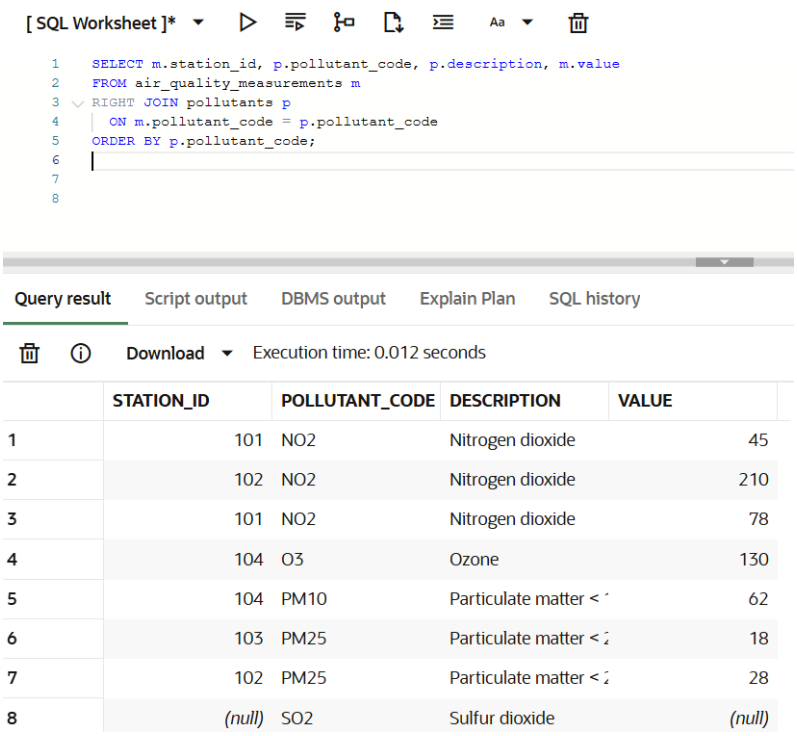
Slika 18. Rezultat upita LEFT JOIN

Vidljivo je da postaja Warsaw-NewStation nema povezanih mjerenja, ali je ipak uključena u rezultat.

5.3 RIGHT JOIN djeluje u suprotnom smjeru: vraća sve retke desne tablice, čak i ako nema podudaranja u lijevoj. U tom slučaju stupci lijeve tablice popunjavaju se vrijednošću NULL.

U oglednoj bazi RIGHT JOIN može prikazati sva definirana onečišćivala, pa i ona za koja još nisu zabilježena mjerenja. To je korisno pri provjeri potpunosti šifrarnika i pronalaženju nekorištenih kategorija.

RIGHT JOIN odgovara na pitanje: *„Koji zapisi postoje u desnoj tablici i jesu li povezani sa zapisima lijeve?"*



Slika 19. Rezultat upita RIGHT JOIN

U posljednjem retku rezultata RIGHT JOIN vidljivo je onečišćivalo SO2 za koje nije zabilježeno mjerenje ni povezana postaja.

6. Podupiti u SQL-u

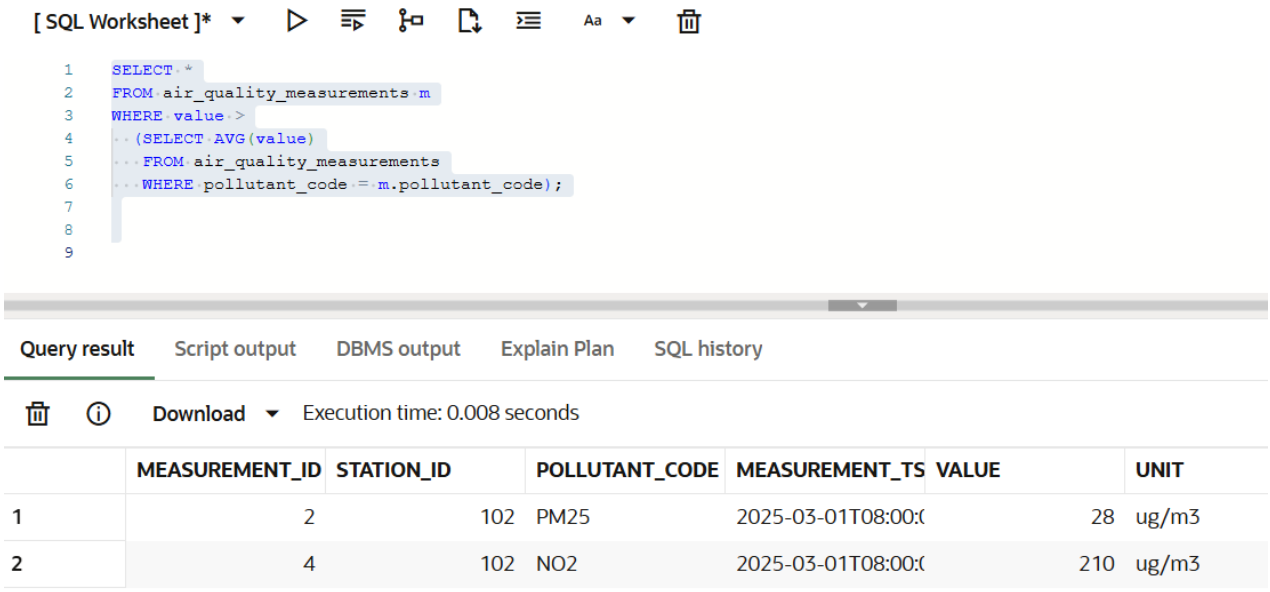
Podupit (engl. subquery) SQL je upit ugniježđen u drugu SQL naredbu. Koristi se kada rezultat jednog upita određuje rezultat drugoga. Tako se složenija logika može izraziti jasno i strukturirano.

U pravilu se podupit vrednuje kako bi se njegov rezultat mogao koristiti u vanjskom upitu. Posebno je koristan pri analizama koje uključuju prosjek, najveću ili najmanju vrijednost te dinamički zadane pragove.

Ovisno o namjeni, podupit može vratiti jednu vrijednost, jedan redak ili skup redaka. Često se koristi za usporedbu pojedinačnih zapisa s agregiranim vrijednostima iz iste tablice.

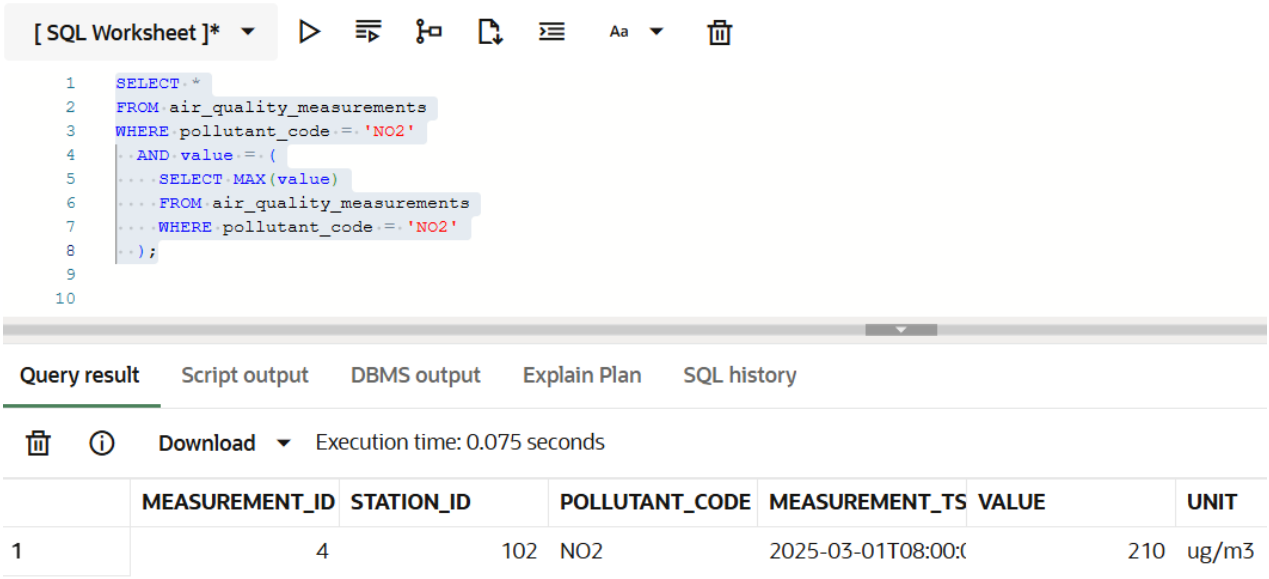
Podupiti mogu biti nekorelirani, odnosno neovisni o vanjskom upitu, ili korelirani, kada ovise o njegovim vrijednostima i vrednuju se za pojedine retke. Obje vrste važne su za analizu relacijskih podataka.

6.1 Izradimo podupit koji prikazuje mjerenja veća od prosjeka za određeno onečišćivalo (slika 20.).



Slika 20. Rezultat izvršavanja podupita

6.2 Podupitom možemo pronaći i postaje s najvećim izmjerenim vrijednostima NO_2 (slika 21.).

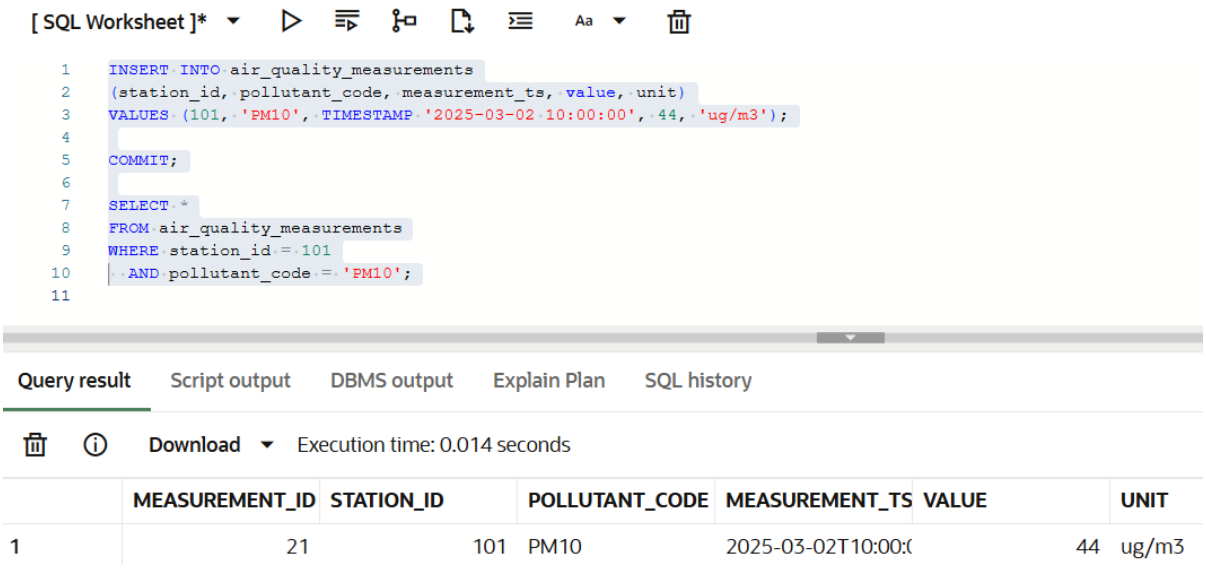


Slika 21. Rezultat izvršavanja podupita

7. Jezik za upravljanje transakcijama (TCL)

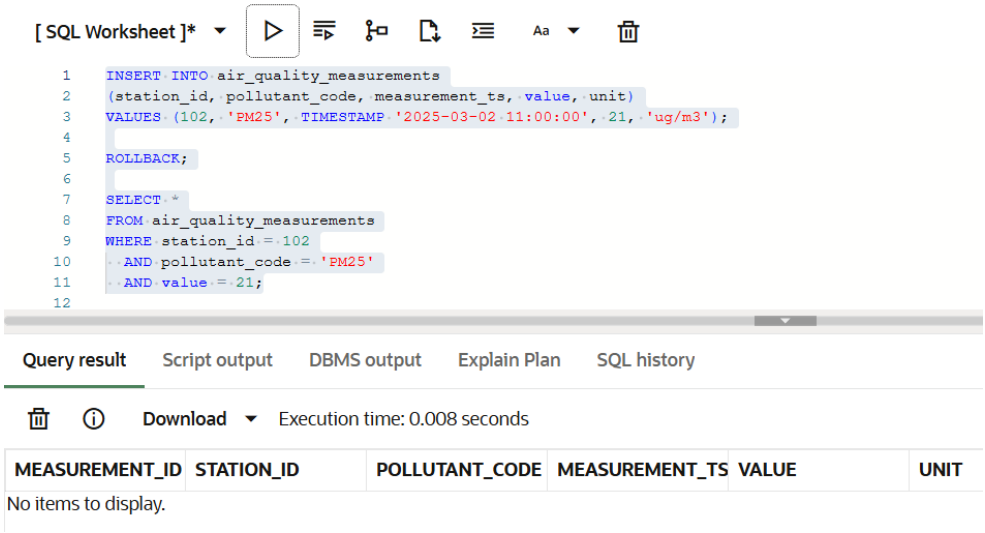
TCL omogućuje pouzdano izvršavanje skupina SQL operacija. U Oracleu transakcije započinju automatski, ali se moraju izričito potvrditi ili poništiti naredbama COMMIT odnosno ROLLBACK. Naredba SAVEPOINT omogućuje djelomično poništavanje transakcije.

7.1 Unesite novo mjerenje kvalitete zraka i trajno spremite promjenu.



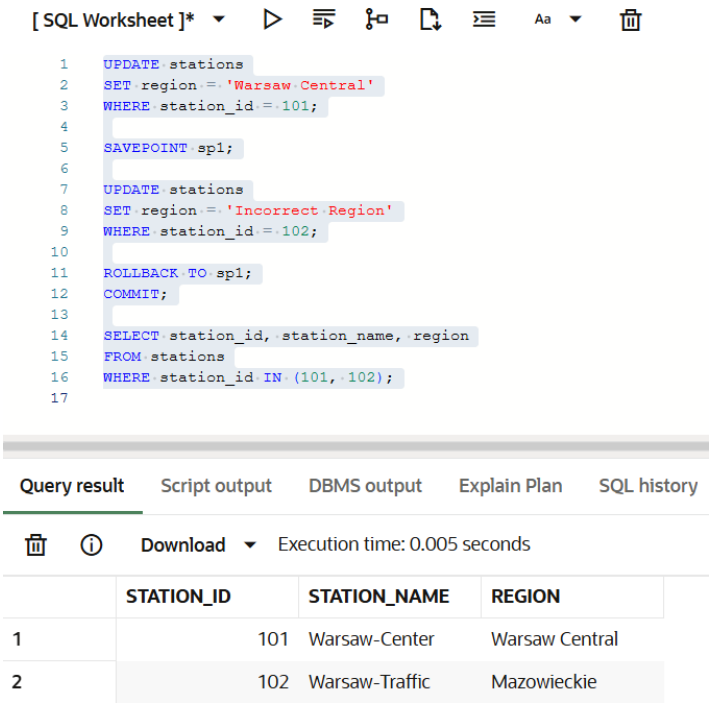
Slika 22. Provjera potvrđene transakcije

7.2 Unesite mjerenje, a zatim poništite operaciju naredbom ROLLBACK.



Slika 23. Provjera poništavanja transakcije

7.3 Provedite više ažuriranja i poništite samo dio transakcije.



Slika 24. Provjera djelomičnog poništavanja transakcije

Analitički zadaci

1. Prikažite sva mjerenja kvalitete zraka, uključujući identifikator postaje, oznaku onečišćivala, izmjerenu vrijednost i vremensku oznaku. Sortirajte rezultate prema vremenu mjerenja.
2. Prikažite sva mjerenja čija je vrijednost veća od 50. Uključite identifikator postaje, onečišćivalo, vrijednost i vrijeme.
3. Dohvatite sva mjerenja NO_2, sortirana od najveće do najmanje vrijednosti.
4. Za svako onečišćivalo pronađite postaju s najvećom izmjerenom vrijednošću. Prikažite naziv postaje, onečišćivalo i vrijednost.
5. Prikažite samo kombinacije postaje i onečišćivala za koje je prosječna vrijednost veća od 40.
6. Za svako onečišćivalo pronađite najveću zabilježenu vrijednost mjerenja.

Praktične vježbe

Vježba 1 – Izrada nove sheme baze podataka

Za analizu skupa podataka India Urban Noise Monitoring (https://www.kaggle.com/datasets/rohanrao/noise-monitoring-data-in-india) studenti trebaju izraditi shemu baze pomoću DDL naredbi. Izradite dvije relacijske tablice: STATIONS (Station, Name, City, State, Type) i STATION_MONTH (Station, Year, Month, Day, Night, DayLimit, NightLimit). Atribut Station definirajte kao primarni ključ tablice STATIONS. Zatim u tablici STATION_MONTH definirajte strani ključ tako da STATION_MONTH.Station upućuje na STATIONS(Station), čime se osigurava referencijalni integritet mjerenja i podataka o postajama.

Vježba 2 – Unos oglednih podataka

Budući da Oracle FreeSQL ne podržava izravan uvoz CSV datoteka, za učitavanje podataka koristimo naredbe INSERT. Podaci su pripremljeni u SQL skriptama stations_insert.sql i station_month_insert.sql. Izvršite skripte kako biste popunili tablice: najprije unesite podatke o postajama u STATIONS, a zatim mjesečna mjerenja u STATION_MONTH.*stations_insert.sqlstation_month_insert.sql*

Vježba 3 – Osnovni upiti

Napišite jednostavne SELECT upite za istraživanje podataka. Najprije prikažite sve postaje iz odabranoga grada (npr. Delhija). Zatim prikažite mjesečna mjerenja buke za odabranu postaju i godinu. Upiti trebaju pomoći razumjeti strukturu i značenje obiju tablica prije složenijih analiza.

Vježba 4 – Filtriranje logičkim operatorima

Dohvatite i usporedite dvije skupine prekoračenja. Najprije vratite sve zapise u kojima su prekoračene obje granice, odnosno Day > DayLimit AND Night > NightLimit. Zatim dohvatite zapise u kojima je prekoračena barem jedna granica: Day > DayLimit OR Night > NightLimit. Objasnite kako izbor logičkog operatora utječe na značenje i veličinu rezultata.

Vježba 5 – Agregiranje

Za svaku postaju izračunajte prosječnu dnevnu i prosječnu noćnu razinu buke. Rezultate grupirajte prema atributu Station. Vježba služi utvrđivanju uporabe agregatnih funkcija i izraza GROUP BY u analitičkom izvješćivanju.

Vježba 5 – Upiti JOIN

Sastavite objedinjeno izvješće koje povezuje podatke o postajama i mjesečna mjerenja. Primjenom INNER JOIN između STATIONS i STATION_MONTH prikažite naziv i grad postaje, vrstu postaje, godinu, mjesec te dnevnu i noćnu razinu buke. Vježba pokazuje kako relacijske veze omogućuju izvješća koja objedinjuju opisne i mjerne podatke.

Provjera znanja

Kviz (5 pitanja): kratka provjera pojmova prije praktičnog rada.

1. Koja tvrdnja najbolje opisuje LEFT JOIN?

A. Vraća samo retke koji se podudaraju u objema tablicama

B. Vraća sve retke lijeve tablice i podudarne retke desne tablice

C. Vraća sve retke desne tablice i podudarne retke lijeve tablice

D. Kombinira sve retke obiju tablica bez uvjeta

2. Koja anomalija nastaje kada brisanje posljednjeg mjerenja postaje uzrokuje gubitak njezinih metapodataka?

A. Anomalija ažuriranja

B. Anomalija umetanja

C. Anomalija brisanja

D. Anomalija spajanja

3. Koja SQL naredba pripada jeziku za definiranje podataka (DDL)?

A. SELECT

B. INSERT

C. DROP

D. UPDATE

4. Koji je od navedenih primjera valjana uporaba podupita u SQL-u?

A. Izračun prosječne vrijednosti onečišćivala unutar uvjeta WHERE

B. Brisanje baze podataka

C. Dodjeljivanje prava pristupa

D. Izrada novih tablica

5. Kojom se klauzulom povezuju redci dviju tablica prema uvjetu podudaranja?

A. GROUP BY

B. ORDER BY

C. JOIN … ON

D. HAVING

Kriteriji vrednovanja: ispravnost SQL upita, potpunost rezultata, jasnoća tumačenja i rezultat kviza.

Component	Weight	Criteria
Kviz	20%	Razumijevanje pojmova
SQL zadaci	50%	Ispravnost, učinkovitost i obuhvat
Izvješće	30%	Tumačenje, jasnoća i vizualizacija

Literatura

Resursi Oracle Academy Cloud

Dokumentacija standarda EU-a o kvaliteti zraka

Referentna dokumentacija za Open-Meteo API

Odabrane knjige otvorenog pristupa o skladištima podataka

Odabrani znanstveni radovi otvorenog pristupa o skladištima podataka u znanosti o okolišu

C. J. Date. Introduction to Database Systems. Addison-Wesley Longman, Inc., 2004. 983 p.

H. Garcia-Molina, J. Ullman, J. Widom. Database Systems: The Complete Book. Pearson, 2008. 1248 p.

P. Rob, C. Coronel. Database Systems: Design, Implementation, and Management. Course Technology, 2011. 728 p.