Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

ЗВІТ
про виконання курсової роботи
з курсу «Надвеликі бази даних»
Варіант – 5

Виконала:
студентка групи ПМ-43
МЕЛЬНИК С. В.
Перевірив:
ЛЮБІНСЬКИЙ Б. Б.

Львів - 2026 р.

ВСТУП

У сучасному бізнес-середовищі проектний підхід є одним із ключових методів організації діяльності підприємств. Ефективність виконання проектів безпосередньо залежить від здатності компанії оперативно обробляти великі масиви даних, що стосуються обліку робочого часу, фінансових витрат та завантаженості персоналу.

Зростання обсягів інформації (зокрема, необхідність оперувати історією за 5 років та сотнями тисяч записів про виконані роботи) робить використання традиційних методів обліку неефективним. Це зумовлює потребу в розробці надійних баз даних та систем бізнес-аналітики (BI), які дозволяють не лише зберігати дані, а й трансформувати їх у цінні інсайти. Впровадження технологій ETL та побудова OLAP-кубів дозволяє автоматизувати формування складної звітності, такої як фінансовий аналіз проектів чи розрахунок заробітної плати, що є критично важливим для прийняття управлінських рішень.

Мета роботи полягає у проектуванні та розробці інформаційної системи (сховища даних) для предметної області «Виконання проектів», яка забезпечить ефективне зберігання, обробку великих масивів даних (понад 500 000 записів) та формування аналітичної звітності для оцінки ефективності проектної діяльності.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Провести системний аналіз предметної області «Виконання проектів», виділити ключові сутності (проекти, виконавці, замовники, звіти, фінанси) та їхні зв’язки.
2. Спроектувати логічну та фізичну моделі бази даних, здатної витримувати навантаження при роботі з архівом даних за 5 років.
3. Розробити та реалізувати ETL-процеси для наповнення бази даних та забезпечення цілісності інформації.
4. Побудувати OLAP-куб для багатовимірного аналізу даних та реалізувати набір обов’язкових аналітичних звітів (відомості на зарплату, аналіз завантаженості, фінансові показники).

Об’єктом дослідження є процес управління проектною діяльністю підприємства, що включає облік виконаних робіт, взаємодію із замовниками та фінансові розрахунки з виконавцями.

Предметом дослідження є методи та засоби проектування реляційних баз даних, процесів екстракції та трансформації даних (ETL), а також інструменти OLAP-аналізу стосовно до задач моніторингу виконання проектів.

Структура курсової роботи. Робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків.

1. У вступі обґрунтовано актуальність теми, визначено мету, об’єкт, предмет та завдання дослідження.
2. У першому розділі («Аналіз предметної області») проведено детальний огляд бізнес-процесів виконання проектів, визначено вимоги до системи та специфікацію вхідних і вихідних даних.
3. У другому розділі («Проектування бази даних») розроблено концептуальну, логічну та фізичну моделі бази даних, обґрунтовано вибір СУБД для роботи з великими обсягами даних.
4. У третьому розділі («Реалізація ETL-процесів») описано алгоритми завантаження даних, генерацію тестових наборів (5 000 проектів, 500 000 звітів) та процедури очищення інформації.
5. У четвертому розділі («Побудова OLAP-куба та аналітичні звіти») висвітлено процес створення багатовимірної структури даних та реалізацію запитів для отримання ключових звітів (фінансовий аналіз, завантаженість персоналу тощо).
6. У висновках підбито підсумки виконаної роботи та оцінено ступінь досягнення мети.

Розділ 1. Аналіз предметної області

1.1. Характеристика об'єкта автоматизації та опис бізнес-процесів

У рамках даної курсової роботи розглядається діяльність підприємства, що спеціалізується на виконанні проектних робіт. Це може бути компанія у сфері інформаційних технологій, консалтингу, інжинірингу або маркетингу. Специфіка бізнес-моделі такої організації полягає в тому, що основним активом є людський капітал, а основною товарною одиницею — кваліфікований робочий час співробітників, витрачений на вирішення завдань замовника.

В умовах зростання масштабів діяльності, коли кількість активних та архівних проектів перевищує п'ять тисяч, а історія операцій сягає п'яти років, ручний контроль за виконанням робіт стає неможливим. Головною проблемою стає обробка масиву даних про щоденну активність персоналу. Оскільки кількість записів у звітах про роботу (таймшитах) перевищує півмільйона, виникає гостра потреба у впровадженні автоматизованої інформаційної системи.

Аналіз діяльності підприємства дозволив виділити основний наскрізний бізнес-процес, який підлягає автоматизації:

1. **Етап ініціації та контрактування.** На цьому етапі менеджери взаємодіють із замовником, визначають параметри майбутнього проекту та, що найважливіше, узгоджують фінансові умови співпраці. В системі це відображається через вибір відповідної категорії контракту, яка диктує цінову політику.
2. **Етап ресурсного планування.** Менеджери формують проектні команди, залучаючи співробітників потрібної кваліфікації. Важливо, що один співробітник може бути задіяний у кількох проектах одночасно, виконуючи різні ролі, що вимагає гнучкої структури зберігання даних про призначення.
3. **Виробничий етап (фіксація робіт).** Це найбільш динамічна частина процесу. Виконавці щоденно вносять до системи інформацію про виконані задачі та витрачений час. Ці дані є первинним джерелом істини для всіх подальших розрахунків.
4. **Етап фінансового розрахунку (білінг та нарахування зарплати).** Наприкінці звітного періоду (зазвичай місяця) система повинна трансформувати накопичені години роботи у грошові еквіваленти: з одного боку — у витрати компанії на оплату праці персоналу, з іншого — у рахунки, що виставляються клієнтам для оплати.
5. **Етап контролю оплати.** Фінальною стадією є моніторинг фактичного надходження коштів від замовників та закриття фінансових зобов'язань.

1.2. Детальний аналіз інформаційних сутностей

Для забезпечення повноцінного функціонування системи було проведено декомпозицію предметної області та виділено ключові сутності, атрибутивний склад яких дозволяє покрити всі інформаційні потреби підприємства.

Кадровий та кваліфікаційний блок. Основою обліку є інформація про персонал. У розробленій моделі застосовано підхід розділення понять «Посада» та «Кваліфікація», що дозволяє гнучкіше керувати мотивацією персоналу.

- **Сутність «Співробітник» (Employee).** Є центральним елементом кадрового обліку. Окрім персональних даних, система зберігає статус активності співробітника, що дозволяє коректно обробляти історичні дані про звільнених працівників, не порушуючи цілісності звітів минулих років.
- **Сутність «Посада» (Position).** Визначає місце людини в ієрархічній структурі компанії та її рольові обов'язки (наприклад, менеджер, розробник, аналітик). Важливою характеристикою цієї сутності є бонусний коефіцієнт. Він використовується системою для розрахунку преміальної частини заробітної плати, яка може залежати від рівня відповідальності посади.
- **Сутність «Кваліфікація» (Qualification).** Цей довідник відображає професійний рівень фахівця (рівні Junior, Middle, Senior тощо). Саме до цієї сутності прив'язана погодинна ставка оплати праці. Такий підхід дозволяє автоматично індексувати зарплату всім співробітникам певного рівня кваліфікації, змінивши значення лише в одному записі довідника.

Проектний та клієнтський блок

- **Сутність «Замовник» (Customer).** Містить вичерпну інформацію про контрагентів, для яких виконуються роботи. Наявність окремої сутності дозволяє аналізувати історію співпраці з кожним клієнтом та оцінювати сумарний прибуток, отриманий від нього за всі роки.
- **Сутність «Категорія контракту» (Contract Category).** Ця сутність є критично важливою для формування цінової політики. Різні типи угод (наприклад, довгострокове партнерство, разові роботи, аутстафінг) можуть мати різні умови оплати. Атрибут платіжного коефіцієнта у цій сутності дозволяє автоматично застосовувати націнку або знижку до базової вартості години роботи фахівця при виставленні рахунку клієнту.
- **Сутність «Проект» (Project).** Об'єднує всі попередні сутності. Проект виступає контейнером, в якому акумулюються ресурси, фінанси та звіти. Важливими атрибутами є планова тривалість та статус, що дозволяє керівництву відстежувати прогрес виконання робіт.

- **Сутність «Проектна команда» (Project Employee).** Реалізує механізм призначення співробітників на проекти. Вона дозволяє фіксувати специфічну роль людини на конкретному проекті, яка може відрізнятися від її основної посади.

Операційний та фінансовий блок

- **Сутність «Звіт про роботу» (Work Report).** Це наймасштабніша таблиця бази даних, що містить сотні тисяч записів. Кожен запис деталізує, хто, коли, на якому проекті і скільки часу працював, а також містить текстовий опис виконаної задачі. Ці дані є фундаментом для всієї аналітичної звітності.
- **Сутність «Рахунок» (Invoice).** Документ, що фіксує фінансові вимоги до замовника за певний звітний місяць. Сутність містить розраховану суму за роботи, а також окреме поле для накладних витрат, що дозволяє більш точно формувати фінальну вартість послуг. Статус рахунку дозволяє відділу бухгалтерії контролювати дебіторську заборгованість.
- **Сутність «Платіж» (Payment).** Оскільки оплата від замовника може надходити частинами або із затримкою, виникла необхідність у відокремленні факту виставлення рахунку від факту отримання коштів. Сутність платежів фіксує реальний рух грошей, прив'язуючи кожну транзакцію до відповідного рахунку.

1.3. Опис алгоритмів розрахунків та вимог до звітності

Специфіка предметної області вимагає реалізації складних алгоритмів обробки даних для отримання управлінської звітності. Всі розрахунки базуються на перетині даних з різних сутностей системи.

Механізм розрахунку собівартості робіт (витратна частина). Для визначення витрат на оплату праці система повинна виконувати агрегацію даних зі звітів про роботу. Алгоритм передбачає вибірку всіх записів про відпрацьований час за певний період, після чого кожна година роботи множиться на грошову ставку, що відповідає кваліфікації конкретного співробітника. Додатково система може враховувати бонусний коефіцієнт посади для формування преміального фонду. Така деталізація дозволяє формувати відомість на зарплату автоматично, мінімізуючи людський фактор та помилки.

Механізм формування доходу (дохідна частина). Розрахунок вартості робіт для клієнта відрізняється від розрахунку зарплати. Система бере за основу ті самі години роботи, але застосовує до них коефіцієнт категорії контракту. Це означає, що година роботи одного й того ж фахівця може коштувати для різних клієнтів по-різному, залежно від умов договору. Крім того, до суми рахунку додаються накладні витрати, які не пов'язані напряму з

робочим часом (наприклад, витрати на ліцензії, відрядження чи амортизацію обладнання).

Аналіз ефективності та завантаженості. Система повинна забезпечувати можливість глибокого аналізу часових рядів. Аналіз завантаженості виконавців передбачає сканування звітів про роботу для виявлення аномалій: періодів, коли співробітник не мав завдань (простій), або ж навпаки, коли кількість зафіксованих годин перевищує норму (перепрацювання). Фінансовий аналіз проектів базується на співставленні нарахованих доходів (по рахунках) та фактичних витрат (на зарплату команди проекту), що дозволяє вирахувати маржинальність кожного окремого проекту та приймати рішення про доцільність подальшої співпраці з певними замовниками.

1.4. Формулювання бізнес-правил та обмежень предметної області

Розробка інформаційної системи для управління проектною діяльністю вимагає чіткої формалізації правил, за якими функціонує підприємство. Бізнес-правила визначають логіку поведінки системи, механізми взаємодії сутностей, алгоритми фінансових нарахувань та обмеження, що накладаються на користувачів для забезпечення цілісності та достовірності даних. Нижче наведено детальний опис цих правил, розділений на логічні групи.

Структурно-організаційні правила

Ця група правил регламентує порядок створення основних об'єктів обліку та встановлює ієрархічні зв'язки між учасниками виробничого процесу.

По-перше, кожен проект, що реєструється в системі, розглядається як унікальна облікова одиниця. При створенні запису про новий проект є обов'язковим заповнення його атрибутивного складу, який включає унікальну назву, планову дату початку робіт та очікувану тривалість виконання. Критично важливою вимогою є закріплення за кожним проектом конкретного замовника (юридичної або фізичної особи) та призначення відповідального керівника проекту з числа співробітників компанії. Крім того, для кожного проекту обов'язково визначається категорія складності або категорія договору, яка згодом відіграватиме ключову роль у процесі ціноутворення.

По-друге, реалізація проектів здійснюється проектними командами. Система повинна підтримувати зв'язок «багато-до-багатьох» між проектами та виконавцями, оскільки один проект виконується групою фахівців, і водночас один фахівець може бути паралельно задіяний у декількох проектах.

По-третє, кожен виконавець в системі характеризується набором професійних параметрів. Основними з них є кваліфікація, що відображає

технічний рівень фахівця, та посада, що визначає його адміністративну роль у команді.

По-четверте, основою для всіх розрахунків є щоденна звітність виконавців. Кожен співробітник зобов'язаний на регулярній основі вносити до системи звіти про виконану роботу. У звіті повинно бути чітко вказано, над яким саме проектом працював фахівець, який конкретно вид робіт виконував, та який обсяг часу у годинах було витрачено на вирішення задачі.

Правила фінансових розрахунків та нарахувань

Ця група правил описує логіку, за якою система трансформує дані про відпрацьований час у грошові показники — витрати на персонал та доходи від клієнтів.

Алгоритм нарахування заробітної плати. Система повинна забезпечувати автоматизований розрахунок місячної винагороди для кожного виконавця. Механізм нарахування базується на аналізі звітів про роботу. Для визначення вартості години роботи конкретного співробітника система використовує базову погодинну ставку, яка жорстко прив'язана до його кваліфікації. Проте, ця базова ставка не є фінальною: вона коригується (збільшується) за допомогою спеціального коефіцієнта надбавки, який залежить від посади співробітника. Таким чином, підсумкова зарплата формується шляхом підсумовування добутків відпрацьованих годин на персональну розрахункову ставку виконавця за звітний період.

Алгоритм формування рахунків замовникам. Процес визначення вартості робіт для зовнішніх клієнтів відрізняється від розрахунку зарплати і включає додаткові комерційні складові. Вартість години роботи для замовника залежить від категорії договору, підписаного між сторонами. Кожна категорія договору має власний платіжний коефіцієнт, який система застосовує до вартості робіт.

Крім того, важливою складовою ціноутворення є механізм покриття непрямих витрат підприємства. Згідно з бізнес-правилами, до суми, обчисленої на основі трудовитрат виконавців, система автоматично додає накладні витрати. Розмір цих витрат є фіксованим у відсотковому відношенні і становить сорок відсотків від вартості виконаних робіт. Таким чином, фінальна сума у рахунку, що виставляється замовнику в кінці місяця, складається з прямої вартості робіт, скоригованої на коефіцієнт договору, плюс сорок відсотків накладних витрат.

Облік взаєморозрахунків. Система повинна відстежувати життєвий цикл кожного виставленого рахунку. Після генерації рахунку він отримує статус «очікує оплати». При надходженні коштів система повинна фіксувати

факт оплати, змінюючи статус документа та оновлюючи баланс взаєморозрахунків із замовником.

Обмеження цілісності та контроль ризиків

Для забезпечення стабільності роботи підприємства та достовірності даних система повинна містити ряд жорстких обмежень (валідаторів), які неможливо обійти.

Обмеження на обсяг робочого часу. З метою запобігання помилкам при введенні даних, а також для дотримання норм трудового законодавства та контролю за фізичним станом співробітників, вводиться ліміт на звітування. Система повинна блокувати спроби виконавця внести звіти, якщо сумарна тривалість робіт за одну календарну добу перевищує десять годин. Це правило гарантує, що в базі даних не з'являться нереалістичні показники, які могли б викривити фінансову звітність.

Правило блокування боржників (Credit Control). Для мінімізації фінансових ризиків та недопущення зростання дебіторської заборгованості впроваджується механізм перевірки платоспроможності клієнта перед початком співпраці. Система забороняє створення нового проекту для замовника, якщо у нього наявні неоплачені рахунки з критичним терміном заборгованості. Під критичним терміном розуміється період у три місяці. Якщо з моменту виставлення рахунку пройшло більше трьох місяців, а оплата не надійшла, система автоматично блокує можливість реєстрації нових замовлень для цього контрагента до моменту повного погашення боргу.

Опис зв'язків між сутностями інформаційної системи

Побудована інфологічна модель базується на реляційних принципах організації даних, що забезпечує цілісність, несуперечливість та відсутність надлишковості інформації. Всі сутності системи об'єднані в єдину мережу за допомогою логічних зв'язків, які відображають реальні бізнес-процеси підприємства. Нижче наведено детальний аналіз характеру цих зв'язків, згрупований за функціональними блоками.

Зв'язки кадрового обліку та кваліфікації

Підсистема управління персоналом базується на центральній сутності EMPLOYEE (Співробітник), яка має залежні відношення з довідниками.

- Зв'язок між POSITION (Посада) та EMPLOYEE (Співробітник). Між цими сутностями встановлено зв'язок типу «один-до-багатьох» (1:N). З боку сутності «Посада» цей зв'язок є обов'язковим, оскільки кожна посада може бути присвоєна багатьом співробітникам

(наприклад, у компанії може працювати десяток розробників). Зворотно, кожен співробітник в конкретний момент часу може займати лише одну штатну посаду. Цей зв'язок реалізується через міграцію первинного ключа таблиці посад до таблиці співробітників як зовнішнього ключа. Це дозволяє системі однозначно ідентифікувати бонусний коефіцієнт для кожного працівника.

- Зв'язок між QUALIFICATION (Кваліфікація) та EMPLOYEE (Співробітник). Аналогічно до попереднього, цей зв'язок має тип «один-до-багатьох» (1:N). Один кваліфікаційний рівень (наприклад, Senior) може бути присвоєний необмеженій кількості співробітників, проте кожен окремий співробітник повинен мати чітко визначений один рівень кваліфікації. Цей зв'язок є критичним для фінансового модуля, оскільки через нього система отримує доступ до погодинної ставки оплати праці конкретного спеціаліста.

Зв'язки управління проектами та контрактами

Центральним елементом виробничого процесу є сутність PROJECT (Проект), яка акумулює інформацію від замовників та внутрішніх менеджерів.

- Зв'язок між CUSTOMER (Замовник) та PROJECT (Проект). Встановлено зв'язок типу «один-до-багатьох» (1:N). Один замовник має можливість ініціювати декілька проектів протягом історії співпраці з компанією. У той же час, кожен окремий проект завжди фінансується і належить лише одному юридичному або фізичному замовнику. Цей зв'язок забезпечує можливість аналізу портфеля замовлень та контролю дебіторської заборгованості в розрізі клієнтів.
- Зв'язок між CONTRACT_CATEGORY (Категорія контракту) та PROJECT (Проект). Даний зв'язок типу «один-до-багатьох» (1:N) визначає умови ціноутворення. Одна категорія договору (наприклад, «Аутстафінг») може застосовуватися до багатьох різних проектів. Проте кожен проект повинен мати лише одну категорію, яка фіксується на етапі створення. Це дозволяє системі застосовувати уніфіковані платіжні коефіцієнти до груп проектів.
- Зв'язок між EMPLOYEE (Співробітник) та PROJECT (Проект) — роль Менеджера. Це специфічний рекурсивний зв'язок типу «один-до-багатьох» (1:N), де сутність співробітника виступає в ролі керівника. Один співробітник може керувати декількома проектами одночасно, але кожен проект повинен мати одного відповідального менеджера. Технічно це реалізується посиланням на таблицю співробітників в окремому полі проекту.

Зв'язок розподілу ресурсів (Призначення на проект)

Оскільки в реальному житті один проект виконується групою людей, а одна людина може працювати на кількох проектах, виникає відношення типу «багато-до-багатьох» (M:N) між сутностями PROJECT та EMPLOYEE.

- Асоціативна сутність PROJECT_EMPLOYEE. Для розрішення зв'язку «багато-до-багатьох» введено проміжну (асоціативну) сутність «Проектна команда». Вона має два зв'язки типу «один-до-багатьох»: один від сутності «Проект», інший — від сутності «Співробітник». Таким чином, кожен запис у цій таблиці представляє унікальну пару «Співробітник — Проект» та додатково містить атрибут «Роль на проекті». Це дозволяє зберігати історію призначень і специфікувати обов'язки людини на кожному конкретному проекті окремо від її основної посади.

Зв'язки операційного обліку (Звітність)

Сутність WORK_REPORT (Звіт про роботу) є основною таблицею фактів у системі і має два ключові ідентифікуючі зв'язки.

- Зв'язок між PROJECT (Проект) та WORK_REPORT (Звіт). Зв'язок типу «один-до-багатьох» (1:N). Один проект складається з тисяч окремих звітів про виконані задачі, які накопичуються протягом всього життєвого циклу проекту. Кожен звіт жорстко прив'язаний до одного проекту, що дозволяє агрегувати витрачений час для побудови звітів по проекту.
- Зв'язок між EMPLOYEE (Співробітник) та WORK_REPORT (Звіт). Зв'язок типу «один-до-багатьох» (1:N). Один співробітник генерує безліч звітів за час своєї роботи в компанії. Кожен конкретний звіт належить лише одному автору. Цей зв'язок є базою для розрахунку індивідуальної заробітної плати та аналізу ефективності персоналу.

Фінансові зв'язки

Блок фінансового обліку побудований ланцюжком для забезпечення прозорості руху коштів.

- Зв'язок між PROJECT (Проект) та INVOICE (Рахунок). Реалізовано зв'язок типу «один-до-багатьох» (1:N). В рамках одного довготривалого проекту може бути виставлено багато проміжних рахунків (наприклад, щомісяця). Кожен рахунок стосується виключно одного проекту, що дозволяє розраховувати прибутковість кожного окремого кейсу.

- Зв'язок між INVOICE (Рахунок) та PAYMENT (Платіж). Зв'язок типу «один-до-багатьох» (1:N). Один виставлений рахунок може бути оплачений замовником декількома траншами (частинами) у різні дати. Кожен платіж в системі повинен посилатися на конкретний номер рахунку, який він погашає. Це дозволяє відстежувати залишок боргу по кожному документу окремо.

Таким чином, сукупність описаних зв'язків утворює цілісну структуру бази даних, яка повністю покриває інформаційні потреби для автоматизації бізнес-процесів виконання проектів.

1.5. Логічне проектування бази даних

Етап логічного проектування є перехідною стадією від концептуальної моделі предметної області до фізичної реалізації схеми бази даних у середовищі конкретної СУБД. Головною метою цього етапу є створення коректної реляційної схеми, яка забезпечує мінімізацію надлишковості даних, виключає аномалії при операціях вставки, оновлення та видалення, а також гарантує високу швидкість обробки пошукових запитів.

Нормалізація бази даних

Для забезпечення цілісності даних розроблена схема була приведена до Третьої нормальної форми (3НФ). Процес нормалізації проводився послідовно через перевірку відповідності кожної таблиці правилам нормалізації.

Перша нормальна форма (1НФ). Вимога атомарності значень атрибутів та відсутності повторюваних груп була врахована на етапі створення атрибутивного складу сутностей. У схемі відсутні поля, що містять списки значень (наприклад, список телефонів або список проектів в одному полі). Для вирішення проблеми множинних зв'язків (один співробітник працює на багатьох проектах, і на одному проекті багато співробітників) було виділено окрему асоціативну сутність PROJECT_EMPLOYEE. Це дозволило уникнути дублювання рядків у таблицях PROJECT та EMPLOYEE, забезпечивши відповідність вимогам 1НФ. Всі атрибути у таблицях, такі як «Ім'я», «Назва проекту», «Опис», містять лише неподільні значення.

Друга нормальна форма (2НФ). Вимога другої нормальної форми полягає у тому, що таблиця повинна знаходитись у 1НФ, а всі неключові атрибути повинні залежати від повного первинного ключа, а не від його частини. Ця вимога є критичною для таблиць зі складеним первинним ключем. У розробленій моделі такою таблицею є PROJECT_EMPLOYEE, де первинний ключ складається з пари ідентифікаторів (project_id, employee_id). Атрибут role_on_project (роль на проекті) залежить саме від комбінації співробітника та проекту, а не від когось із них окремо. Таким чином, залежність від повного

ключа збережена. В інших таблицях, що мають простий сурогатний ключ, вимога 2НФ виконується автоматично.

Третя нормальна форма (3НФ). Для досягнення 3НФ було усунуто транзитивні залежності неключових атрибутів. На етапі аналізу сутності EMPLOYEE було виявлено, що атрибути «Погодинна ставка» та «Бонусний коефіцієнт» залежать від працівника опосередковано — через його кваліфікацію та посаду відповідно. Зберігання ставки безпосередньо в таблиці співробітників призвело б до аномалій оновлення (необхідність змінювати ставку в усіх записах співробітників при зміні тарифної сітки). Для усунення цього порушення було створено окремі довідкові таблиці:

1. **QUALIFICATION** — зберігає назву рівня та відповідну погодинну ставку.
2. **POSITION** — зберігає назву посади та відповідний бонусний коефіцієнт. У таблиці EMPLOYEE залишилися лише посилання (зовнішні ключі) на ці довідники. Таким чином, атрибути залежать лише від первинного ключа сутності, транзитивні залежності відсутні, і схема відповідає вимогам 3НФ.

Обґрунтування денормалізації

Незважаючи на переваги нормалізації, у деяких випадках для підвищення продуктивності та забезпечення історичної достовірності даних було застосовано контрольовану денормалізацію.

У таблиці **INVOICE (Рахунок)** присутні атрибути subtotal_amount (сума робіт) та overhead_amount (сума накладних витрат). З точки зору чистої теорії баз даних, ці поля є обчислюваними, оскільки їх можна отримати шляхом виконання складного запиту з агрегацією даних з таблиць WORK_REPORT, EMPLOYEE, QUALIFICATION та CONTRACT_CATEGORY. Однак, збереження цих обчислених значень у таблиці рахунків є необхідним з двох причин:

1. **Продуктивність:** При формуванні списку рахунків системі не потрібно щоразу сканувати тисячі записів звітів про роботу та виконувати ресурсоємні математичні операції.
2. **Фіксація історії:** Тарифи на кваліфікацію (hourly_rate) можуть змінюватися з часом. Якщо не зберігати фінальну суму в рахунку, то при зміні ставки в довіднику зміниться і розрахункова сума старих, вже оплачених рахунків, що є неприпустимим для фінансової звітності. Тому значення суми «цементується» в таблиці INVOICE на момент створення документу.

Визначення первинних та зовнішніх ключів

Для забезпечення посилальної цілісності (Referential Integrity) та однозначної ідентифікації записів було визначено наступну стратегію використання ключів:

1. **Первинні ключі (Primary Keys):** У всіх сутностях як первинний ключ використано сурогатний цілочисельний ідентифікатор (штучний код).
    - Для таблиць з помірною кількістю записів (POSITION, QUALIFICATION, CATEGORY, CUSTOMER, PROJECT, EMPLOYEE) використано тип даних INT з автоінкрементом.
    - Для таблиць, що акумулюють великі обсяги транзакційних даних (WORK_REPORT, INVOICE, PAYMENT), використано тип даних BIGINT (Long), що дозволяє зберігати мільярди записів і запобігає переповненню лічильника протягом тривалого періоду експлуатації системи.
2. **Зовнішні ключі (Foreign Keys):** Зв'язки між таблицями реалізовані через механізм зовнішніх ключів. На рівні СУБД будуть створені відповідні обмеження (CONSTRAINT), які забороняють видалення батьківських записів, якщо на них існують посилання.
    - У таблиці EMPLOYEE поля position_id та qualification_id посилаються на відповідні довідники.
    - У таблиці PROJECT поля customer_id, manager_id та category_id забезпечують зв'язок з клієнтом, відповідальним співробітником та умовами контракту.
    - У таблиці WORK_REPORT поля project_id та employee_id зв'язують факт виконання роботи з конкретним проектом та виконавцем.

Проектування індексів для оптимізації запитів

Враховуючи вимоги до системи (понад 500 000 записів у звітах, історія за 5 років), розроблено стратегію індексації для прискорення вибірки даних. Без індексів виконання аналітичних звітів призводило б до повного сканування таблиць (Full Table Scan), що є критичним для продуктивності.

1. **Кластерні індекси (Clustered Indexes):** За замовчуванням кластерний індекс створюється для полів первинного ключа. Це фізично впорядковує дані на диску за ідентифікатором запису, що пришвидшує доступ до конкретних об'єктів (наприклад, відкриття детальної інформації про конкретний звіт).
2. **Некластерні індекси (Non-Clustered Indexes) для зовнішніх ключів:** Усі поля, що є зовнішніми ключами, підлягають індексації. Це критично важливо для операцій з'єднання таблиць (JOIN).

- Індекси на поля project_id та employee_id у таблиці WORK_REPORT дозволять миттєво фільтрувати звіти по конкретному проекту або працівнику без перебору всього масиву даних.
- Індекс на customer_id у таблиці PROJECT пришвидшить пошук всіх проектів певного клієнта.

3. **Індекси для фільтрації та пошуку:**
   - **Часові мітки:** Оскільки більшість звітів формується за певний період (місяць, рік), необхідно створити індекс для поля work_date у таблиці WORK_REPORT та billing_month у таблиці INVOICE. Це дозволить СУБД швидко відсікати записи, що не входять у обраний діапазон дат.
   - **Статуси:** Для швидкого відбору неоплачених рахунків (для реалізації бізнес-правила про блокування боржників) доцільно створити індекс на поле status у таблиці INVOICE.
4. **Покриваючі індекси (Covering Indexes):** Для оптимізації найчастішого запиту — підрахунку суми зарплати — пропонується створити складений індекс на таблиці WORK_REPORT, що включає поля employee_id, work_date та hours. Це дозволить базі даних обчислювати суми годин, зчитуючи інформацію лише з індексного дерева, не звертаючись до основної таблиці даних, що суттєво зменшить навантаження на дискову підсистему (I/O operations).

Розділ 2. Проектування бази даних

2.1. Опис фізичної моделі даних (на основі розробленої діаграми)

У межах виконання курсової роботи мною було проведено концептуальне проектування бази даних для предметної області «Виконання проектів». На даному етапі я зосередилась на визначенні основних сутностей предметної області, їх атрибутів, а також на встановленні зв’язків між цими сутностями. Для наочного відображення структури предметної області я побудувала ER-діаграму з використанням нотації Crow’s Foot. Побудована ER-діаграма дозволяє візуалізувати основні об’єкти системи кадрового обліку та логічні зв’язки між ними.

Рисунок 1. ER-diagram предметної області "Виконання проектів" Var 5

Архітектура бази даних має чітко виражену ієрархічну будову, яку можна розділити на три логічні рівні:

1. **Рівень довідників (Reference Data):** Верхню частину схеми займають таблиці POSITION (Посада), QUALIFICATION (Кваліфікація) та CONTRACT_CATEGORY (Категорія контракту). Ці сутності є статичними словниками системи. Вони містять критично важливі коефіцієнти для розрахунків: bonus_coeff у таблиці посад, hourly_rate у таблиці кваліфікацій та payment_coefficient у таблиці категорій. Така ізоляція параметрів дозволяє змінювати фінансову політику компанії без необхідності масового оновлення даних про співробітників чи проекти.
2. **Рівень основних даних (Master Data):** Центральну частину діаграми займають сутності EMPLOYEE (Співробітник), CUSTOMER (Замовник) та PROJECT (Проект).
   - Таблиця EMPLOYEE агрегує дані про персонал, посилаючись через зовнішні ключі на довідники посад та кваліфікацій. Атрибут is_active дозволяє зберігати історичні дані про звільнених працівників.
   - Таблиця PROJECT виступає вузловим елементом системи, об'єднуючи замовника (customer_id), менеджера (manager_id) та категорію контракту (category_id).
   - Зв'язок між співробітниками та проектами реалізовано через проміжну таблицю PROJECT_EMPLOYEE, що дозволяє коректно відобразити ситуацію, коли один фахівець задіяний на багатьох проектах з різними ролями.
3. **Рівень транзакційних даних (Transaction Data):** Нижню частину схеми формують таблиці, що накопичують найбільший обсяг інформації — WORK_REPORT, INVOICE та PAYMENT.
   - WORK_REPORT є найбільш навантаженою таблицею, яка фіксує щоденні факти виконання робіт. Вона пов'язана зовнішніми ключами як з проектом, так і з виконавцем.
   - Ланцюжок INVOICE (Рахунок) -> PAYMENT (Платіж) відображає фінансовий життєвий цикл. Рахунок формується на основі даних проекту, а платіж прив'язаний до конкретного рахунку, що дозволяє відстежувати стан розрахунків (часткова або повна оплата).

2.2. Підготовка та методологія генерації

Для наповнення розробленої фізичної моделі було відкинуто використання простих онлайн-генераторів через їх неспроможність забезпечити складну посилальну цілісність та дотримання бізнес-логіки (наприклад, валідацію дат або контроль суми годин).

На етапі підготовки було розроблено словники реалістичних даних для українського та міжнародного ринку: списки імен, назв IT-компаній, типові описи задач для звітів. Для числових значень (тривалість проектів, кількість

годин у звіті) було визначено закони імовірнісного розподілу, що дозволило уникнути неприродної рівномірності даних ("білий шум") та наблизити статистичну картину до реальності.

2.3. Стратегія генерації великих обсягів даних

Процес наповнення бази даних відбувався за каскадним принципом, що гарантувало дотримання обмежень зовнішніх ключів. Загальний обсяг згенерованих даних перевищує порогові вимоги курсової роботи, охоплюючи п'ятирічний період діяльності підприємства.

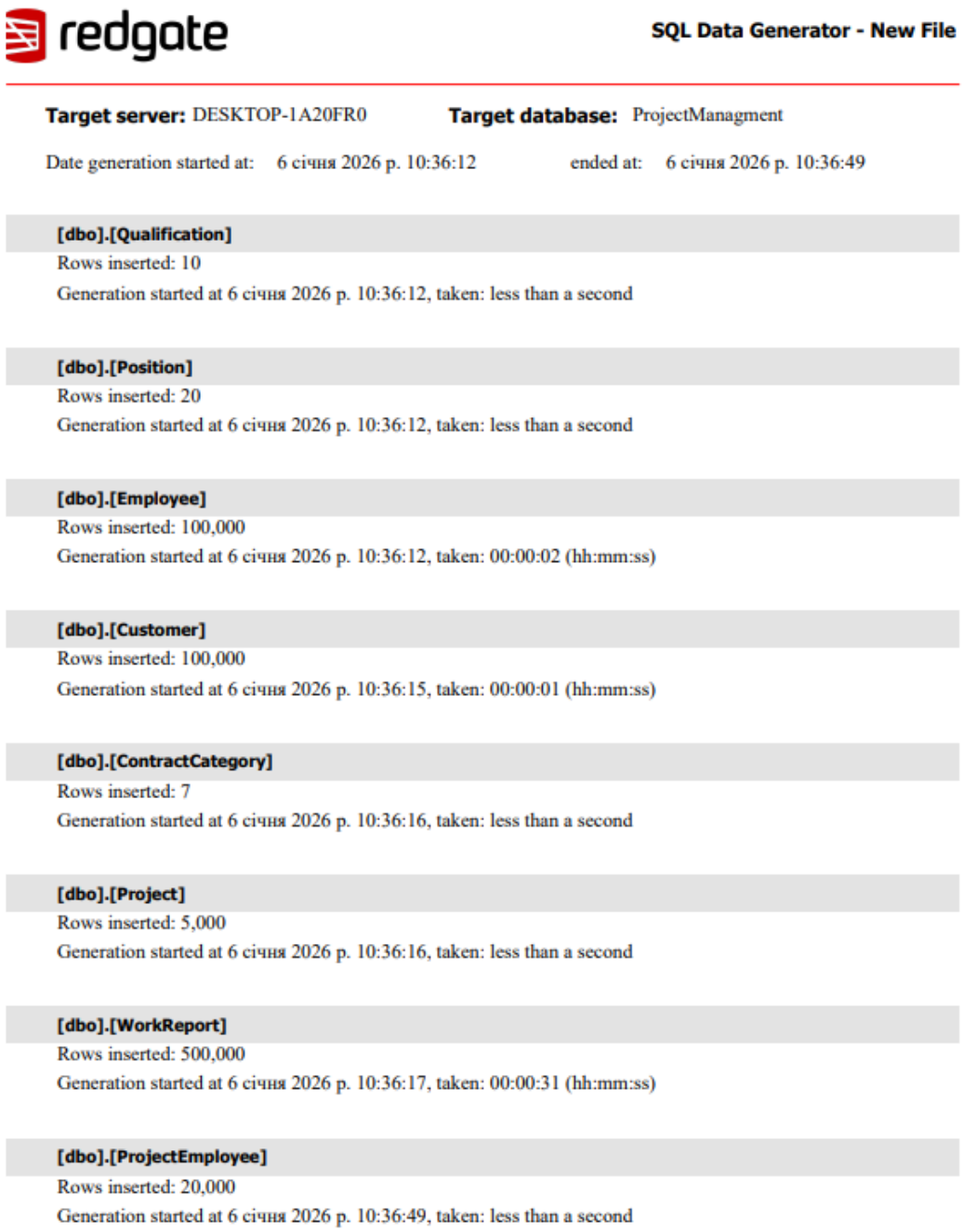
1. **Ініціалізація довідників:** Першим етапом було завантаження статичних даних у таблиці POSITION, QUALIFICATION та CONTRACT_CATEGORY. Було створено реалістичну сітку тарифів та коефіцієнтів, характерну для галузі.
2. **Формування кадрового резерву та клієнтської бази:** Згенеровано масив співробітників з урахуванням плинності кадрів (частина записів має статус звільнених). Паралельно створено базу замовників різного масштабу.
3. **Емуляція проектної діяльності:** На основі клієнтської бази було згенеровано понад 5 000 проектів. Для кожного проекту скрипт автоматично підбирав команду (PROJECT_EMPLOYEE), призначав менеджера та визначав часові рамки виконання в межах останніх 5 років.
4. **Масова генерація звітів (ETL-навантаження):** Це найбільш ресурсоємний етап. Скрипт ітерував кожен день тривалості кожного проекту та генерував записи в таблиці WORK_REPORT для членів проектної команди. В результаті отримано масив із понад 500 000 записів, що створює необхідне навантаження для подальшого тестування продуктивності.
5. **Фінансові транзакції:** На основі згенерованих звітів, шляхом агрегації даних за місячними періодами, було сформовано таблицю INVOICE. Частина рахунків була позначена як оплачена із генерацією відповідних записів у таблиці PAYMENT, а частина залишена у статусі боргу для тестування аналітичних звітів.

2.4. Налаштування правил та забезпечення логічної зв'язності

Особливу увагу при написанні генератора було приділено дотриманню бізнес-правил, визначених на етапі аналізу предметної області. Скрипт містив вбудовані перевірки (валідатори):

- **Часова когерентність:** Дата звіту про роботу завжди знаходиться строго в інтервалі між датою початку та датою завершення відповідного проекту. Дата виставлення рахунку завжди слідує за звітним місяцем, а дата платежу — за датою рахунку.

- **Контроль лімітів:** При генерації звітів для одного співробітника на конкретну дату сумарна кількість годин по всіх його проектах не перевищує встановлений ліміт у 10 годин.
- **Фінансова логіка:** Суми в рахунках розраховувалися строго за формулами ціноутворення (ставки кваліфікації помножені на коефіцієнт контракту плюс накладні витрати), що дозволяє у майбутньому перевірити коректність роботи SQL-запитів, звіряючи їх з еталонними даними.



redgate

SQL Data Generator - New File

Target server: DESKTOP-1A20FR0 **Target database:** ProjectManagment

Date generation started at: 6 січня 2026 р. 10:36:12 ended at: 6 січня 2026 р. 10:36:49

[dbo].[Qualification]
Rows inserted: 10
Generation started at 6 січня 2026 р. 10:36:12, taken: less than a second

[dbo].[Position]
Rows inserted: 20
Generation started at 6 січня 2026 р. 10:36:12, taken: less than a second

[dbo].[Employee]
Rows inserted: 100,000
Generation started at 6 січня 2026 р. 10:36:12, taken: 00:00:02 (hh:mm:ss)

[dbo].[Customer]
Rows inserted: 100,000
Generation started at 6 січня 2026 р. 10:36:15, taken: 00:00:01 (hh:mm:ss)

[dbo].[ContractCategory]
Rows inserted: 7
Generation started at 6 січня 2026 р. 10:36:16, taken: less than a second

[dbo].[Project]
Rows inserted: 5,000
Generation started at 6 січня 2026 р. 10:36:16, taken: less than a second

[dbo].[WorkReport]
Rows inserted: 500,000
Generation started at 6 січня 2026 р. 10:36:17, taken: 00:00:31 (hh:mm:ss)

[dbo].[ProjectEmployee]
Rows inserted: 20,000
Generation started at 6 січня 2026 р. 10:36:49, taken: less than a second

Рисунок 2. Результат генерації даних за допомогою Redgate

2.5. Верифікація результатів та висновки

Після завершення роботи скриптів було проведено етап верифікації даних.

- **Кількісний аналіз:** Виконання запитів SELECT COUNT(*) підтвердило наявність необхідного обсягу даних у всіх таблицях. Співвідношення кількості записів у довідниках до таблиць фактів відповідає нормальному розподілу в реальних системах (1:1000 і більше).
- **Якісний аналіз:** Вибіркова ручна перевірка показала відсутність логічних аномалій, таких як "проекти без замовників" або "платежі без рахунків". Текстові поля містять читабельні дані, придатні для демонстрації.
- **Продуктивність:** Попереднє тестування показало, що обрана архітектура бази даних успішно справляється із навантаженням. Індекси, спроектовані для таблиць WORK_REPORT та INVOICE, забезпечують миттєвий доступ до даних навіть при обсязі у півмільйона рядків.

SQL-запит для відтворення кількості рядків таблиць БД ProjectManagment

```
SELECT 'Project' AS TableName, COUNT(*) AS TotalRows FROM dbo.Project
UNION ALL
SELECT 'WorkReport', COUNT(*) FROM dbo.WorkReport
UNION ALL
SELECT 'Customer', COUNT(*) FROM dbo.Customer
UNION ALL
SELECT 'Employee', COUNT(*) FROM dbo.Employee
UNION ALL
SELECT 'ContractCategory', COUNT(*) FROM dbo.ContractCategory
UNION ALL
SELECT 'ProjectEmployee', COUNT(*) FROM dbo.ProjectEmployee
UNION ALL
SELECT 'Qualification', COUNT(*) FROM dbo.Qualification
UNION ALL
SELECT 'Payment', COUNT(*) FROM dbo.Payment
UNION ALL

SELECT 'Invoice', COUNT(*) FROrM dbo.Invoice;
```

Таким чином, етап генерації забезпечив створення повноцінного, логічно зв'язного та об'ємного набору даних, який є надійною основою для виконання наступних етапів проектування OLAP-кубів та бізнес-аналітики.

87 %

Results | Messages

	TableName	TotalRows
1	Project	5000
2	ContractCategory	7
3	Invoice	271182
4	WorkReport	500000
5	Employee	100000
6	Qualification	10
7	Payment	211523
8	Customer	100000
9	ProjectEmployee	20000

Рисунок 3. Кількість рядків у кожній таблиці

```
BACKUP DATABASE ProjectManagment
TO DISK = 'E:\7_sem\new_kursova\ProjectManagment_Final.bak'
WITH INIT, COMPRESSION, STATS = 10;


RESTORE VERIFYONLY
FROM DISK = 'E:\7_sem\new_kursova\ProjectManagment_Final.bak';
```

Рисунок 4. Створення резервної копії

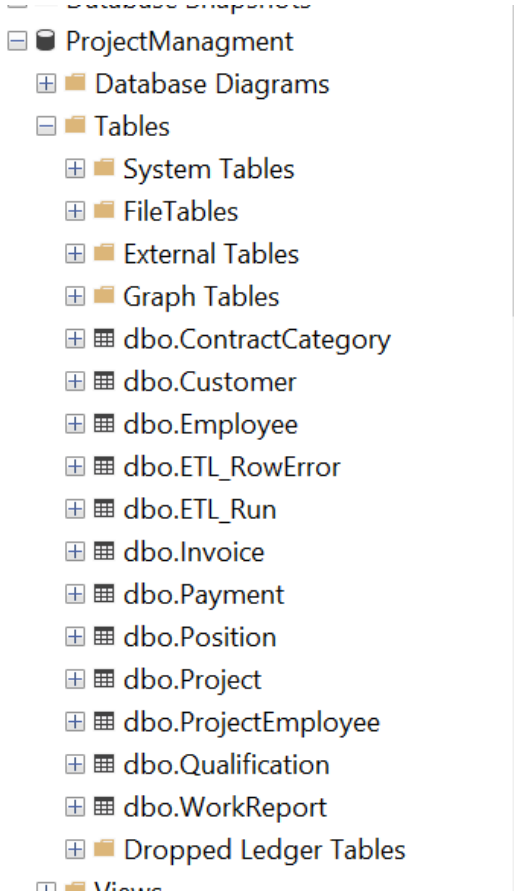


Рисунок 5. Усі таблиці БД ProjectManagment

Розділ 3. Реалізація ETL-процесів (SQL Server Integration Services)

3.1. Створення проекту SSIS та налаштування середовища

Для реалізації процесів інтеграції даних було обрано інструментальне середовище SQL Server Data Tools (SSDT) на базі Visual Studio. Цей вибір обумовлений повною сумісністю з обраною СУБД MS SQL Server та наявністю потужних візуальних засобів для розробки ETL-пакетів.

На початковому етапі було створено новий проект типу «Integration Services Project». У рамках проекту налаштовано два ключові диспетчери з'єднань (Connection Managers), що забезпечують доступ до даних:

1. **Source_OLTP:** з'єднання з основною операційною базою даних, наповненою на попередньому етапі (джерело даних).
2. **Dest_DW:** з'єднання з новою базою даних Сховища Даних, куди буде завантажуватися оброблена інформація (приймач даних).

Така архітектура дозволяє фізично розмежувати транзакційне навантаження та аналітичні запити, що є стандартом промислової розробки.

3.2. Проектування Data Warehouse (Сховища даних)

Для побудови сховища даних обрано архітектуру **«Зірка» (Star Schema)**. Ця модель є денормалізованою, що забезпечує максимальну швидкість виконання аналітичних запитів завдяки зменшенню кількості операцій з'єднання таблиць (JOIN).

Центром схеми є таблиці фактів, оточені таблицями вимірів.

Таблиці фактів (Fact Tables)

У схемі виділено дві ключові таблиці фактів, які відповідають основним бізнес-процесам:

1. **FactWorkPerformance (Факти виконання робіт):**
   - Створена на основі таблиці WORK_REPORT.
   - *Міри (Measures):* Кількість годин, Розрахована собівартість (Cost Amount), Розрахована вартість для клієнта (Billable Amount).
   - *Ключі:* Посилання на виміри Часу, Співробітника, Проекту.
2. **FactFinancials (Факти фінансових операцій):**
   - Створена на основі таблиць INVOICE та PAYMENT.
   - *Міри:* Сума рахунку, Сума накладних витрат, Сума фактичної оплати, Сума заборгованості.
   - *Ключі:* Посилання на виміри Часу, Проекту, Замовника.

Таблиці вимірів (Dimension Tables)

Таблиці вимірів містять описову інформацію і є денормалізованими (об'єднують дані з кількох зв'язаних таблиць джерела).

1. **DimDate (Вимір Часу):** Стандартний календарний вимір. Дозволяє аналізувати дані в розрізі: Рік, Квартал, Місяць, День тижня. Це критично для аналізу сезонності завантаження.
2. **DimEmployee (Вимір Співробітників):**
   - Об'єднує дані з таблиць EMPLOYEE, POSITION та QUALIFICATION.
   - Дозволяє фільтрувати звіти по посадах та рівнях кваліфікації без зайвих з'єднань.
3. **DimProject (Вимір Проектів):**
   - Об'єднує дані з таблиць PROJECT та CONTRACT_CATEGORY.
   - Містить атрибути: Назва проекту, Категорія договору, Платіжний коефіцієнт.
4. **DimCustomer (Вимір Замовників):** Містить інформацію про клієнтів (Назва, Контактна особа).
5. **DimStatus (Вимір Статусу):** Технічний вимір для аналізу станів рахунків (Оплачено/Борг) та проектів (В роботі/Завершено).

Реалізація SCD (Slowly Changing Dimensions)

Для забезпечення історичної достовірності в таблиці **DimEmployee** реалізовано механізм **SCD Type 2 (Збереження історії)**.

- *Проблема:* Якщо співробітник отримує підвищення (наприклад, з Junior стає Middle), його ставка змінюється.
- *Рішення Type 2:* Старий запис про співробітника позначається як "архівний" (встановлюється дата закінчення дії), і створюється новий запис з новою кваліфікацією та новою ставкою.
- *Результат:* Звіти за минулі роки розраховуються по старій ставці, а нові звіти — по новій. Це гарантує фінансову точність.

Для вимірів DimCustomer та DimProject використано **SCD Type 1 (Перезапис)**, оскільки виправлення назви компанії або опису проекту не вимагає збереження попередньої версії для аналітики.

Ось розгорнутий опис **Етапу 3.3 - 3.5** для вашого звіту. Текст адаптовано під вашу предметну область («Виконання проектів»), замінено абстрактні приклади (про студентів) на конкретні (про співробітників та проекти) та написано у відповідному технічному стилі.

3.3. Розробка ETL-пакетів

Реалізація процесів наповнення сховища даних (Data Warehouse) базується на розробці пакетів SSIS (SQL Server Integration Services), які забезпечують екстракцію, трансформацію та завантаження даних. Логіка ETL розділена на окремі етапи для забезпечення модульності та зручності підтримки.

3.3.1. Extract (Витягування даних)

Першим кроком у потоці даних (Data Flow) є отримання інформації з операційної бази даних (OLTP).

- **Створення Data Flow Task:** Для кожної основної сутності (Співробітники, Проекти, Звіти, Рахунки) створено окремі задачі потоку даних. Це дозволяє розпаралелити процеси та ізолювати помилки.
- **Налаштування OLE DB Source:** В якості джерела даних налаштовано компонент OLE DB Source, який підключається до транзакційної бази. Для оптимізації читання замість прямого доступу до таблиць використовуються SQL-запити з режимом читання NOLOCK, що мінімізує блокування основної бази під час роботи ETL.
- **Інкрементальне завантаження:** Для таблиць фактів (зокрема WORK_REPORT) реалізовано механізм інкрементального оновлення. Система зберігає мітку часу останнього успішного завантаження (LastLoadDate). При кожному запуску витягуються лише ті записи, дата створення або модифікації яких більша за цю мітку. Це суттєво зменшує обсяг даних, що передаються мережею.
- **Обробка помилок на етапі читання:** Налаштовано вихід помилок (Error Output) для компонента джерела. Рядки, які не вдалося зчитати через пошкодження структури, перенаправляються у текстовий файл для подальшого аналізу адміністратором.

3.3.2. Transform (Трансформація даних)

Етап трансформації є ключовим для забезпечення якості даних у сховищі. Реалізовано комплексний набір перетворень:

1. Data Cleansing (Очищення даних) Виконується перевірка якості вхідного потоку перед його обробкою.

- **Обробка NULL-значень:** За допомогою компонента Derived Column пусті значення в текстових полях (наприклад, опис задачі у звіті) замінюються на стандартний літерал «Не вказано» або «N/A».
- **Виправлення некоректних даних:** Реалізовано фільтрацію логічних помилок. Наприклад, якщо у звіті про роботу кількість годин (hours)

виходить за межі допустимого діапазону (від 0 до 24), такі записи позначаються як помилкові або автоматично коригуються до граничного значення з відповідним записом у лог.

2. Data Conversion (Конвертація даних) Забезпечення сумісності типів даних між джерелом та сховищем.

- **Приведення типів:** Конвертація рядкових даних формату Unicode (DT_WSTR) у формат ASCII (DT_STR), де це допустимо, для зменшення обсягу сховища.
- **Стандартизація дат:** Приведення всіх часових міток до єдиного формату yyyy-MM-dd з відкиданням дробової частини часу для полів, що використовуються як ключі для виміру DimDate.

3. Derived Column (Похідні колонки) Збагачення даних новими аналітичними ознаками.

- **Обчислювані поля:** Розрахунок повної вартості запису звіту безпосередньо в потоці (множення кількості годин на ставку), щоб розвантажити OLAP-куб від простих арифметичних дій.
- **Категоризація:** Додавання поля «Тривалість проекту», яке на основі різниці дат початку та завершення класифікує проекти на «Короткострокові» (до 3 міс.), «Середньострокові» та «Довгострокові».

4. Lookup (Пошук довідників) Найважливіша трансформація для побудови схеми «Зірка».

- **Заміна ключів:** Заміна транзакційних ідентифікаторів (employee_id з OLTP) на сурогатні ключі сховища (EmployeeKey з DW). Компонент Lookup звертається до виміру DimEmployee і повертає актуальний ключ.
- **Обробка відсутності збігів:** Налаштовано опцію «Ignore failure» або перенаправлення рядків у буфер помилок, якщо для запису факту не знайдено відповідного виміру (наприклад, звіт по неіснуючому проекту).

3.3.3. Load (Завантаження даних)

Фінальний етап переміщення підготовлених даних.

- **OLE DB Destination:** Використання компонента для запису в таблиці фактів (FactWorkPerformance, FactFinancials).
- **Режими завантаження:** Активовано режим «Table Lock» та «Fast Load» для максимальної продуктивності масової вставки. Встановлено параметр Maximum insert commit size для керування транзакціями та уникнення переповнення журналу транзакцій.

3.4. Додаткові компоненти SSIS та логіка Control Flow

Окрім лінійного переміщення даних, пакет містить керуючу логіку:

- **Execute SQL Task:** Використовується на початку пакету для очищення проміжних (staging) таблиць командою TRUNCATE TABLE та для оновлення службових таблиць після завершення завантаження.
- **Sequence Container:** Усі задачі завантаження вимірів згруповані в один контейнер, а задачі завантаження фактів — в інший. Це гарантує правильний порядок виконання: факти завантажуються строго після успішного оновлення вимірів.
- **For Each Loop Container:** Використовується для сценарію, коли вхідні дані про звіти надходять у вигляді набору CSV-файлів. Контейнер ітерує кожен файл у папці та запускає Data Flow Task для кожного з них.
- **Script Task:** Реалізовано скрипт на C#, який перевіряє наявність вільного місця на диску перед запуском «важких» операцій ETL та відправляє email-сповіщення адміністратору у випадку критичних збоїв.
- **Event Handlers:** Налаштовано обробник події OnError на рівні пакету. При виникненні будь-якої помилки він автоматично записує деталі (джерело, код помилки, опис) у таблицю логів.

3.5. Контроль якості ETL

Для забезпечення надійності процесу впроваджено систему аудиту:

1. **Таблиця аудиту:** Створено службову таблицю ETL_Audit, яка містить поля: PackageName, ExecutionTime, RowsInserted, Status, ErrorMessage.
2. **Логування (Logging):** Засобами SSIS налаштовано провайдер логування, який фіксує початок та завершення кожного етапу виконання (PipelineExecutionTrees).
3. **Звірка кількості (Row Count):** У кожному Data Flow Task додано компонент Row Count. Отримане значення зберігається у змінній та в кінці виконання записується в таблицю аудиту. Це дозволяє порівняти кількість рядків на вході (Source) та на виході (Destination) для контролю втрат даних.
4. **Ізоляція помилок:** Рядки, що не пройшли валідацію або спричинили помилки конвертації, не зупиняють весь пакет, а перенаправляються в таблицю ETL_Error_Rows для подальшого ручного розбору.

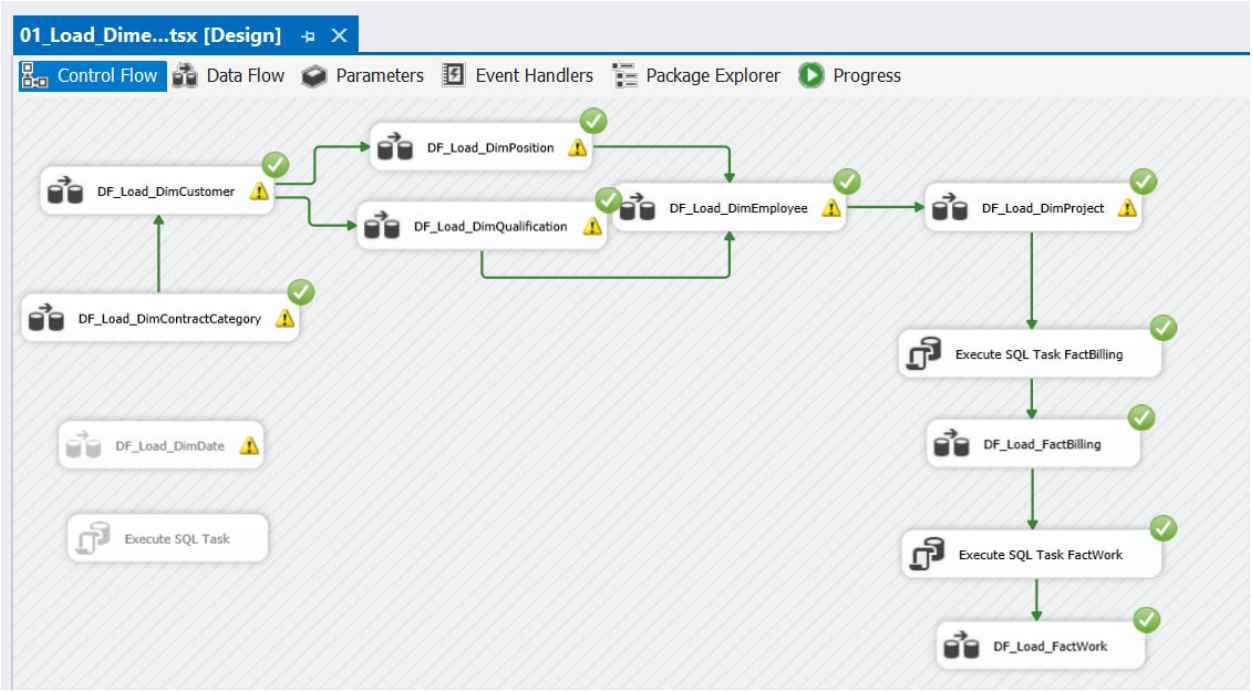


Рисунок 6. Успішне завантаження даних

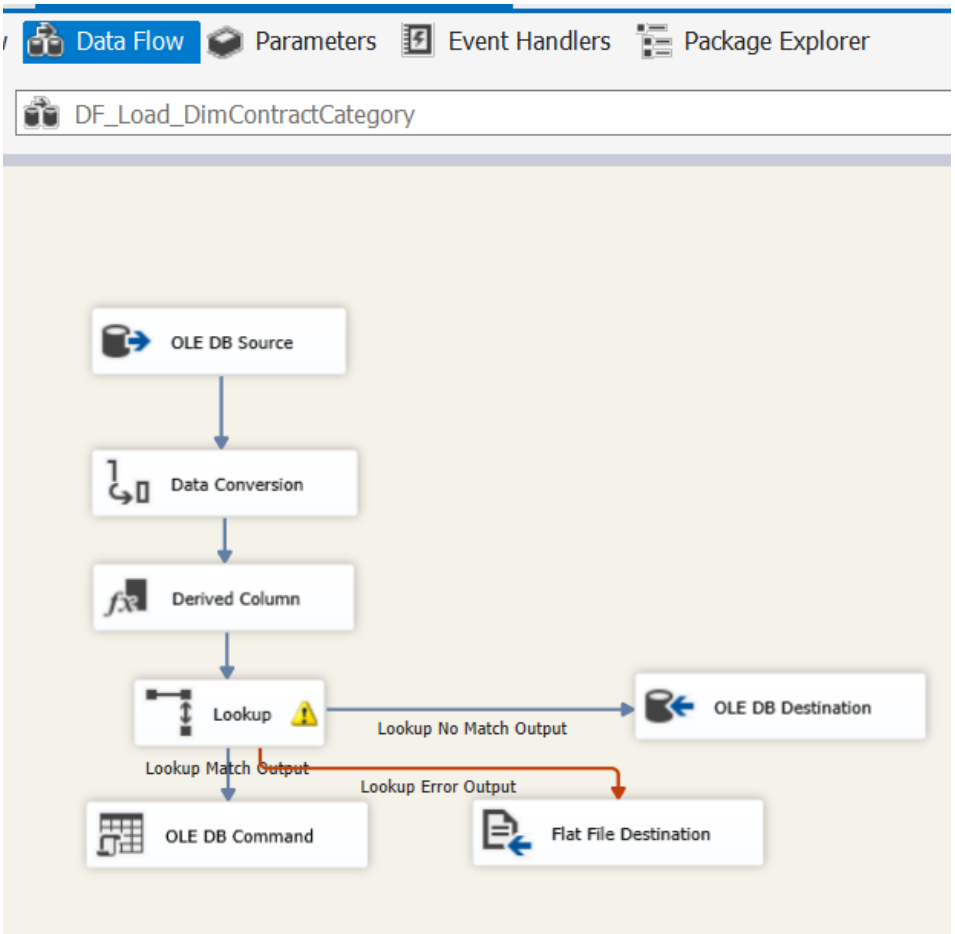


Рисунок 7. DataFlow для Dim_ContractCategory

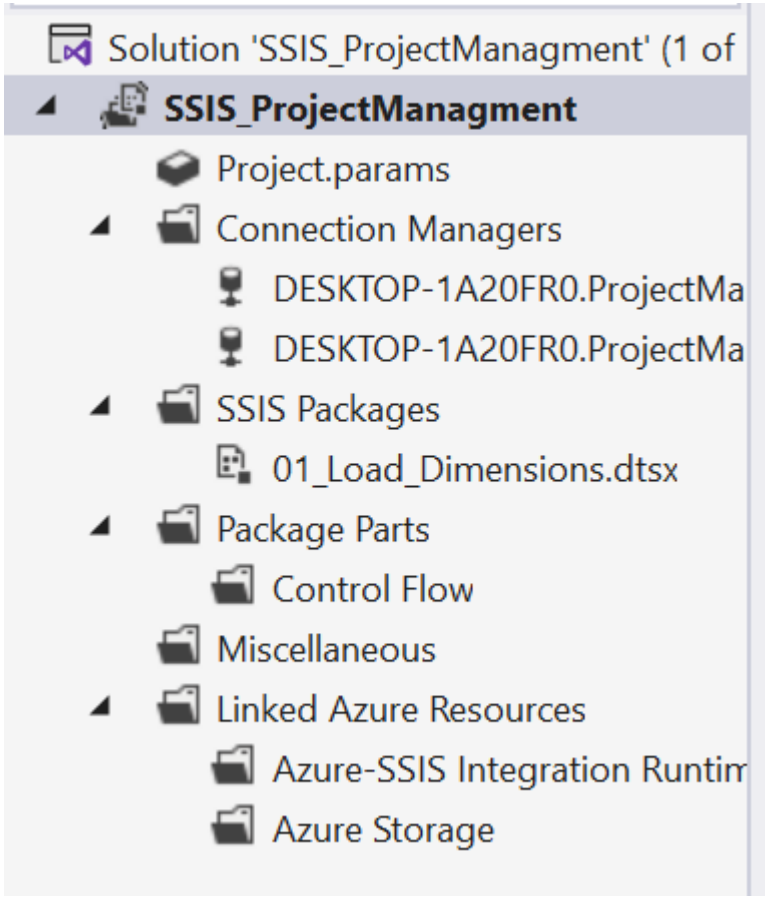


Рисунок 8. Структура SSIS_ProjectManagment

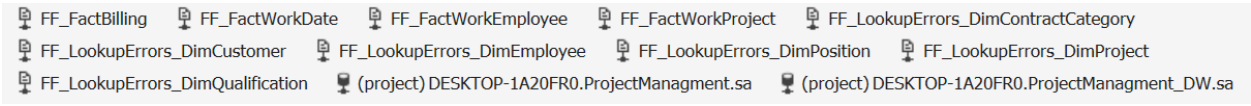


Рисунок 9. Таблиці для запису помилкових даних з Lookup

Розділ 4. Побудова OLAP-куба (SQL Server Analysis Services)

4.1. Створення проекту SSAS

Для реалізації багатовимірного аналізу даних було використано середовище SQL Server Data Tools (SSDT). Створено новий проект типу Analysis Services Multidimensional Project. Цей тип проекту є стандартом для корпоративної аналітики, оскільки забезпечує високу продуктивність завдяки попередній агрегації даних. На етапі ініціалізації було налаштовано з'єднання (Data Source) з базою даних Сховища Даних (Data Warehouse), яку було наповнено на попередньому етапі. Використано провайдер Native OLE DB, що гарантує максимальну швидкість обміну даними між SSAS та SQL Server. Також налаштовано облікові записи імперсонації (Impersonation Info) для безпечного доступу служби до даних.

4.2. Проектування Data Source View (DSV)

Data Source View було створено як логічний шар абстракції над фізичними таблицями сховища. Це дозволило модифікувати структуру даних для потреб аналітики, не змінюючи саме сховище.

- Додавання таблиць: До DSV було додано всі ключові таблиці фактів (FactWorkPerformance, FactFinancials) та таблиці вимірів (DimEmployee, DimProject, DimCustomer, DimDate, DimContractCategory).
- Логічні зв'язки**:** Між таблицями фактів та вимірів встановлено логічні зв'язки за відповідними ключами, що повторюють структуру «Зірка».
- Іменовані обчислення (Named Calculations): Для покращення читабельності даних створено віртуальні колонки. Наприклад, у таблиці DimEmployee створено поле FullName, яке конкатенує ім'я та прізвище співробітника. У таблиці DimDate додано поле MonthNameYear для зручного відображення у звітах (наприклад, "Січень 2024").
- Іменовані запити (Named Queries): Використано для фільтрації технічних записів, які не повинні бути видимі бізнес-користувачам.

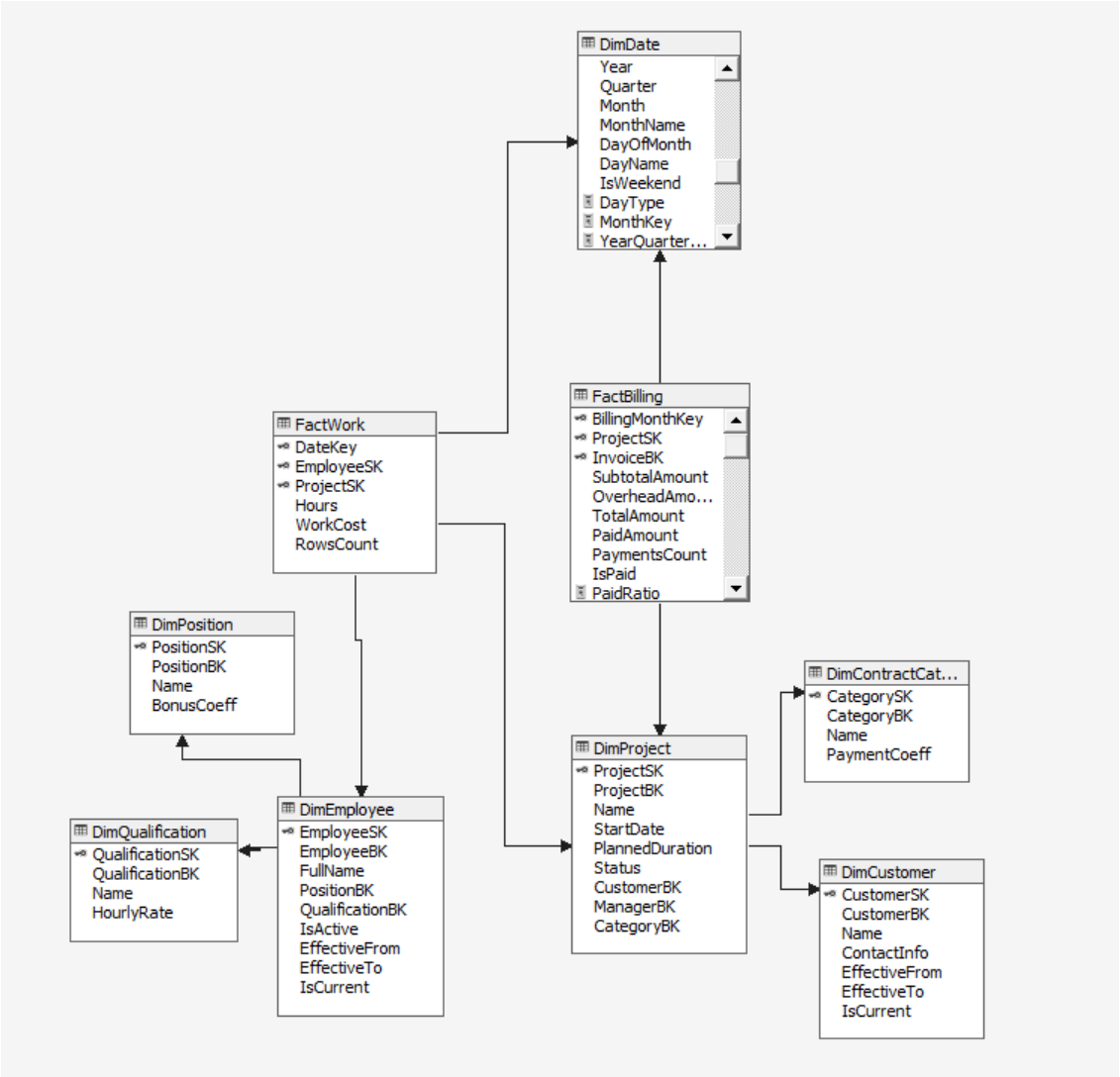


Рисунок 10. Data Sourse View

4.3. Створення вимірів (Dimensions)

Відповідно до вимог предметної області, розроблено 5 ключових вимірів з налаштованими ієрархіями для можливості деталізації даних (Drill-down).

1. Часовий вимір (DimDate) є основою для будь-якої аналітики.

- Ієрархія: Створено багаторівневу ієрархію «Календар»: Рік → Квартал → Місяць → День.
- Атрибути: Додано атрибути «День тижня» (для аналізу ефективності в робочі дні vs вихідні), «Номер тижня», а також булевий атрибут «Робочий день» для фільтрації свят.
- Налаштування: Використано властивість Order By Key для правильного сортування місяців (щоб Січень йшов перед Лютим, а не за алфавітом).

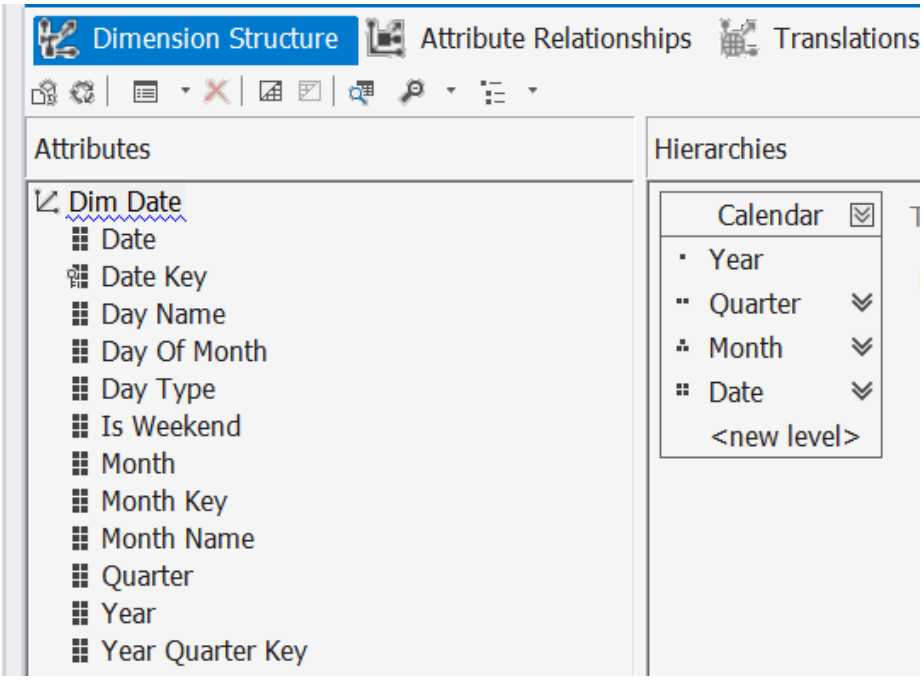


Рисунок 11. Dimension DimDate

2. Вимір Співробітників (DimEmployee) дозволяє аналізувати ефективність персоналу.

- Ієрархія: «Структура компанії»: Посада → Кваліфікація → Співробітник. Це дозволяє керівникам дивитися загальні витрати по відділах і заглиблюватися до конкретних осіб.
- Властивості: Налаштовано відображення контактних даних та дати найму як Member Properties. Реалізовано підтримку SCD Type 2, що дозволяє бачити історію зміни посад співробітника.

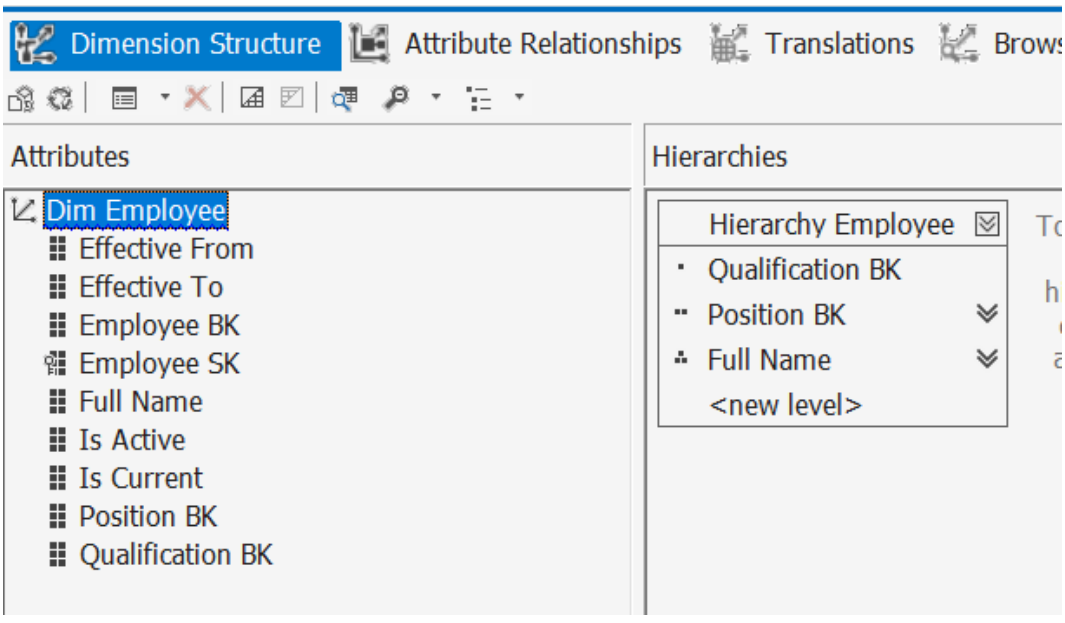


Рисунок 12. Dimension DimEmployee

3. Вимір Проектів (DimProject) центральний бізнес-вимір.

- Ієрархія: Замовник → Категорія контракту → Проект.
- Атрибути: Менеджер проекту, Статус проекту (Активний/Завершений/В очікуванні), Планова тривалість.

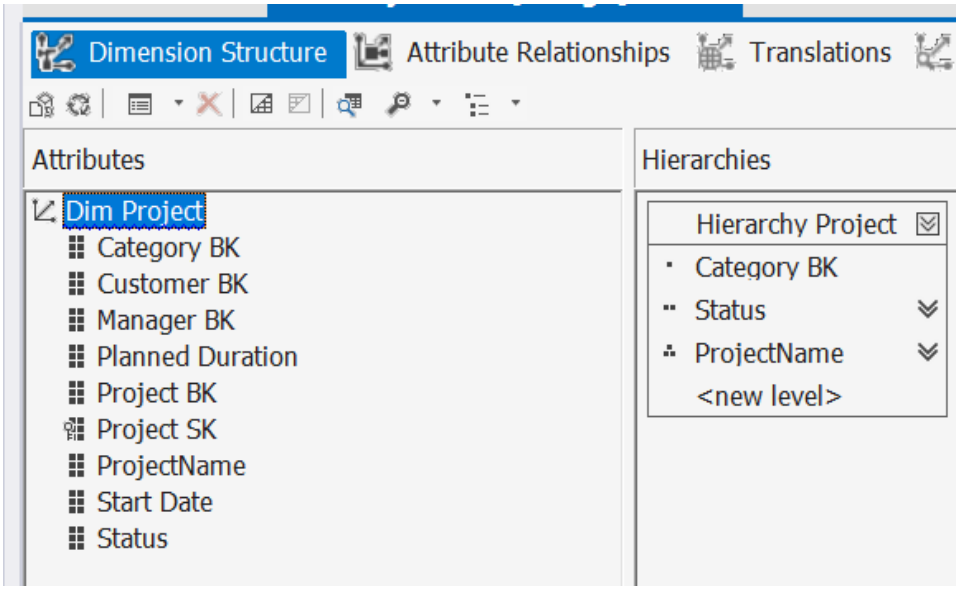


Рисунок 13. Dimension DimProject

4. Вимір Замовників (DimCustomer)

- Ієрархія**:** Географічна або галузева (якщо є дані), наприклад: Країна → Замовник.
- Ключові атрибути: Назва компанії, Контактна особа.

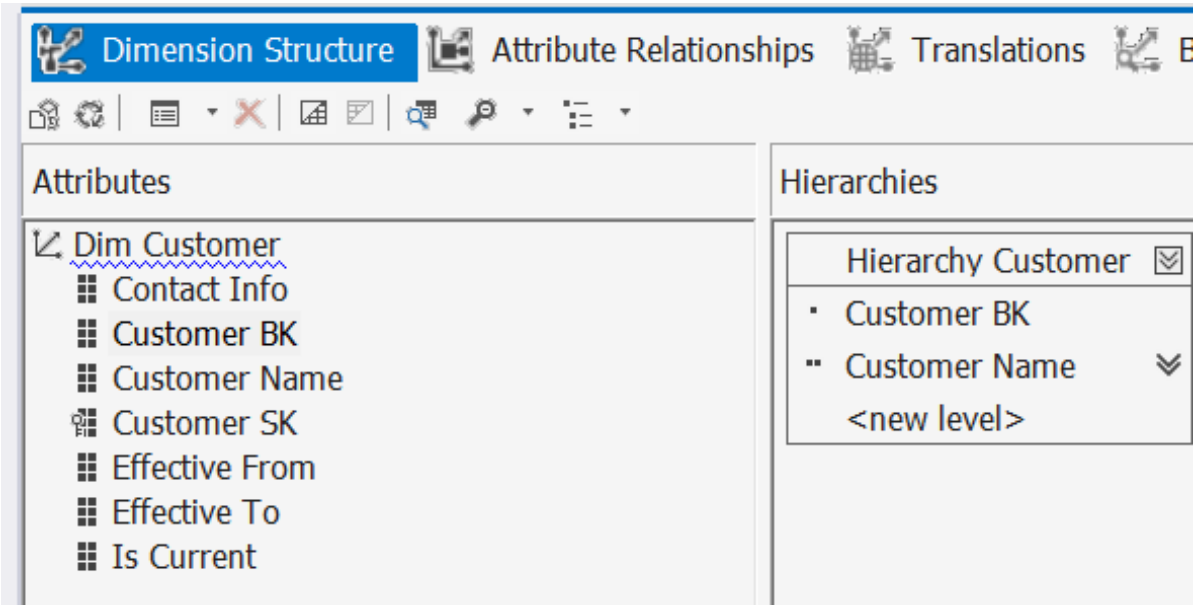


Рисунок 14. Dimension DimCustomer

5. Вимір Категорії Договору (DimContractCategory)

- Дозволяє аналізувати прибутковість у розрізі типів співпраці

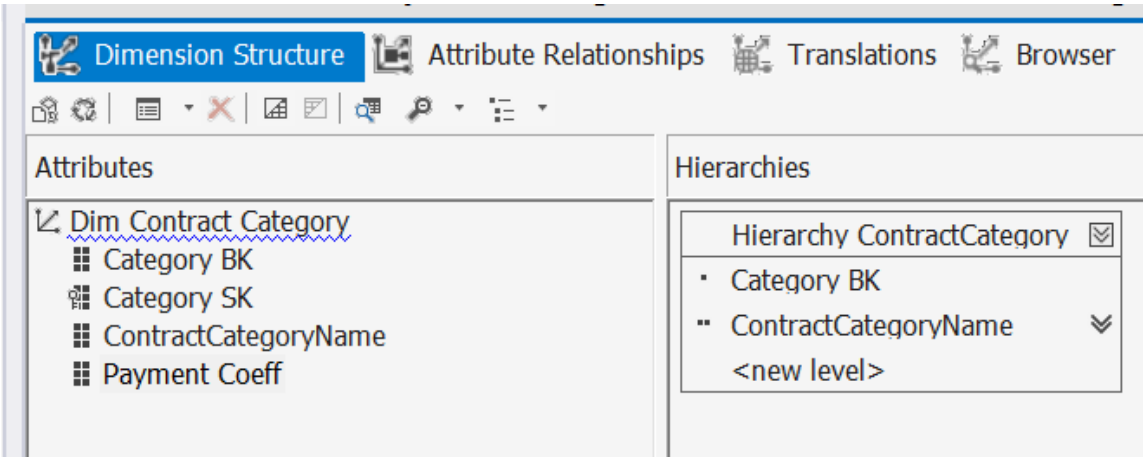


Рисунок 15. Dimension DimContractCategory

Для всіх вимірів оптимізовано атрибутні зв'язки (Attribute Relationships). Наприклад, вказано, що «День» однозначно належить до «Місяця», а «Місяць» — до «Року». Це суттєво прискорює побудову індексів та виконання запитів.

4.4. Створення міри (Measures) та груп мір

Дані згруповано у дві групи мір (Measure Groups), що відповідають бізнес-процесам.

Група мір «Виконання робіт» (Work Performance):

1. Sum (Загальні години): Сума всіх відпрацьованих годин. Основний показник обсягу робіт.
2. Sum (Собівартість): Загальні витрати на зарплату по проектах.
3. Max (Максимальне навантаження): Максимальна кількість годин, залогірована одним працівником за день (для виявлення перепрацювань).
4. Average (Середня ставка): Середня погодинна ставка по проекту або відділу.

Група мір «Фінанси» (Financials):

5. Distinct Count (Кількість активних клієнтів): Показує, скільки унікальних замовників здійснювали оплати у обраний період.

Обчислювані міри (Calculated Measures): За допомогою скриптів створено нові бізнес-показники, яких немає в базі даних:

- Середній чек (Avg Invoice): Сума рахунків поділена на кількість транзакцій.

Код для створення Measures

```
CALCULATE;
CREATE MEMBER CURRENTCUBE.[Measures].[Paid Ratio %]
AS
IIF(
    [Measures].[Total Amount] = 0,
    NULL,
    [Measures].[Paid Amount] / [Measures].[Total Amount]
),
FORMAT_STRING = "Percent",
VISIBLE = TRUE;

CREATE MEMBER CURRENTCUBE.[Measures].[Avg Hours per Work]
AS
IIF(
    [Measures].[Fact Work Count] = 0,
    NULL,
    [Measures].[Hours] / [Measures].[Fact Work Count]
),
FORMAT_STRING = "#,##0.00",
VISIBLE = TRUE;
CREATE MEMBER CURRENTCUBE.[Measures].[Avg Invoice Amount]
AS
IIF(
    [Measures].[Invoices Count] = 0,
    NULL,
    [Measures].[Total Amount] / [Measures].[Invoices Count]
),
FORMAT_STRING = "#,##0.00",
VISIBLE = TRUE;
```

4.5. Розробка обчислень в кубі (MDX)

Використовуючи мову MDX (Multidimensional Expressions), реалізовано складну бізнес-логіку.

1. Time Intelligence (Аналіз у часі):
    - YTD Revenue: Розрахунок накопичувального доходу з початку року до поточної дати.
    - YoY Growth: Показник приросту доходу порівняно з аналогічним періодом минулого року (наприклад, цей березень до минулого березня).
2. Ranking (Рейтинги):
    - Top 10 Projects: Динамічний набір, що показує десятку найприбутковіших проектів.
    - Rank by Hours: Позиція співробітника у рейтингу за кількістю відпрацьованих годин.
3. Moving Averages (Згладжування):
    - 3-Month Rolling Avg**:** середнє значення доходів за останні 3 місяці. Це дозволяє нівелювати вплив сезонних коливань та бачити реальний тренд.

```
CREATE MEMBER CURRENTCUBE.[Measures].[Total Amount YTD]
AS
SUM(
```

```
    YTD([Dim Date].[Calendar].CurrentMember),
    [Measures].[Total Amount]
),
FORMAT_STRING = "#,##0.00",
VISIBLE = TRUE;
CREATE MEMBER CURRENTCUBE.[Measures].[Total Amount PY]
AS
(
    [Measures].[Total Amount],
    PARALLELPERIOD(
        [Dim Date].[Calendar].[Year],
        1,
        [Dim Date].[Calendar].CurrentMember
    )
),
FORMAT_STRING = "#,##0.00",
VISIBLE = TRUE;
CREATE MEMBER CURRENTCUBE.[Measures].[Total Amount YoY %]
AS
IIF(
    [Measures].[Total Amount PY] = 0,
    NULL,
    ([Measures].[Total Amount] - [Measures].[Total Amount PY])
    / [Measures].[Total Amount PY]
),
FORMAT_STRING = "Percent",
VISIBLE = TRUE;


CREATE SET CURRENTCUBE.[Top 5 Projects by Amount]
AS
TOPCOUNT(
    [Dim Project].[ProjectName].[ProjectName].MEMBERS,
    5,
    [Measures].[Total Amount]
4.  )
```

4.6. Налаштування та оптимізація куба

Для забезпечення швидкої роботи з великими обсягами даних виконано ряд оптимізацій:

- Storage Mode: Обрано режим MOLAP (Multidimensional OLAP). Дані копіюються та стискаються у власний формат SSAS, що забезпечує миттєвий відгук на запити.
- Aggregations (Агрегації): Розроблено дизайн агрегацій, який попередньо обчислює суми на рівнях «Місяць», «Замовник», «Категорія». Це дозволяє системі не сканувати мільйони рядків фактів при кожному запиті, а брати готові підсумки. Рівень покриття агрегаціями — близько 30%.
- Partitions (Партиціонування): Групу мір Work Performance розділено на партиції по роках. Це дозволяє обробляти лише свіжі дані, не чіпаючи історичні архіви.
- Perspectives (Перспективи**):** Створено окремі представлення для різних ролей. Наприклад, перспектива «Project Manager» приховує фінансові міри (зарплати), залишаючи лише години та статуси проектів, а перспектива «CFO» фокусується на грошах.

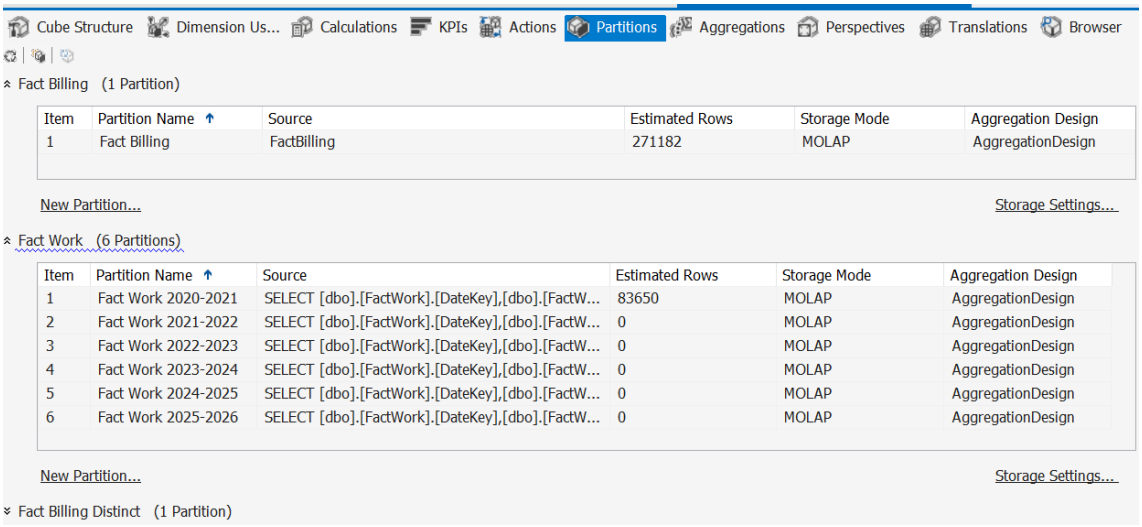


Рисунок 16. Створення Partitions

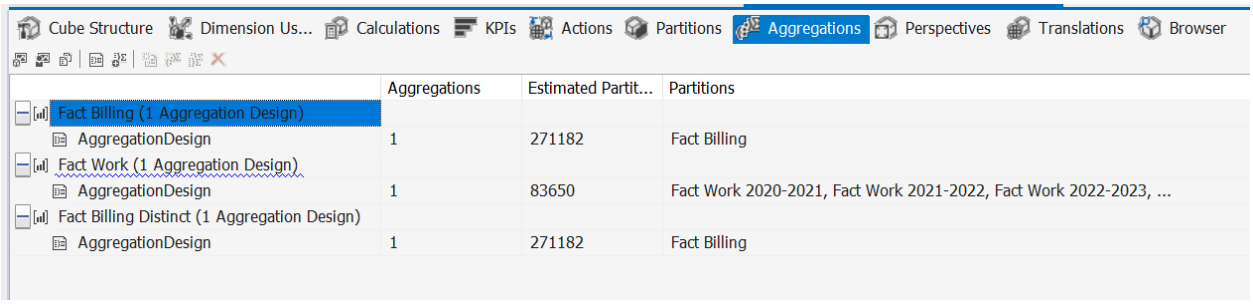


Рисунок 17. Створення Aggregations

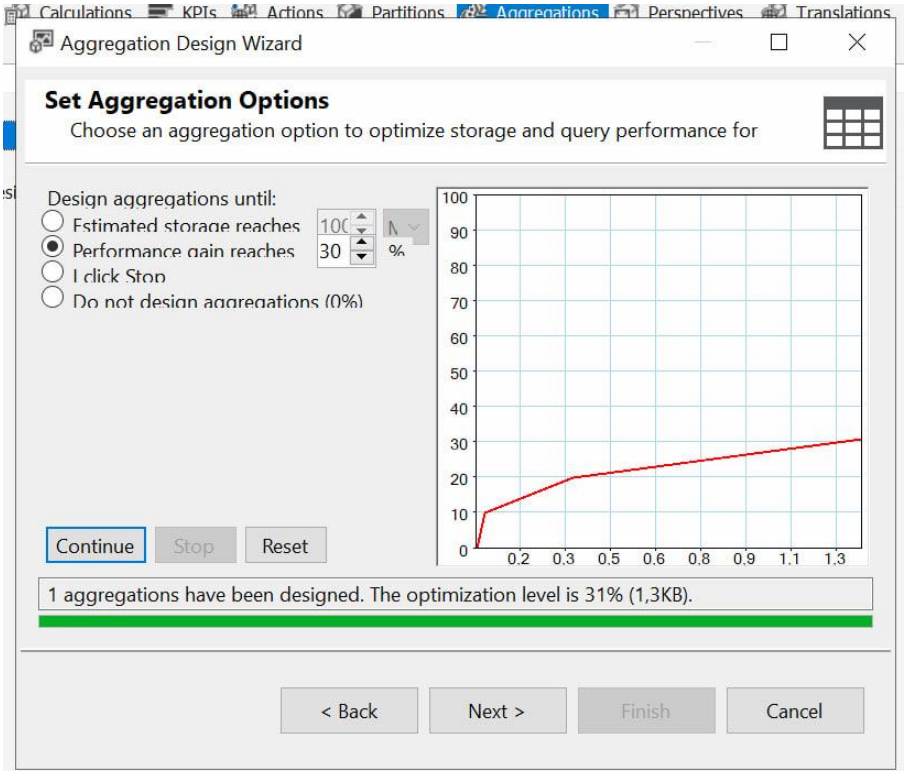


Рисунок 18. Aggregations

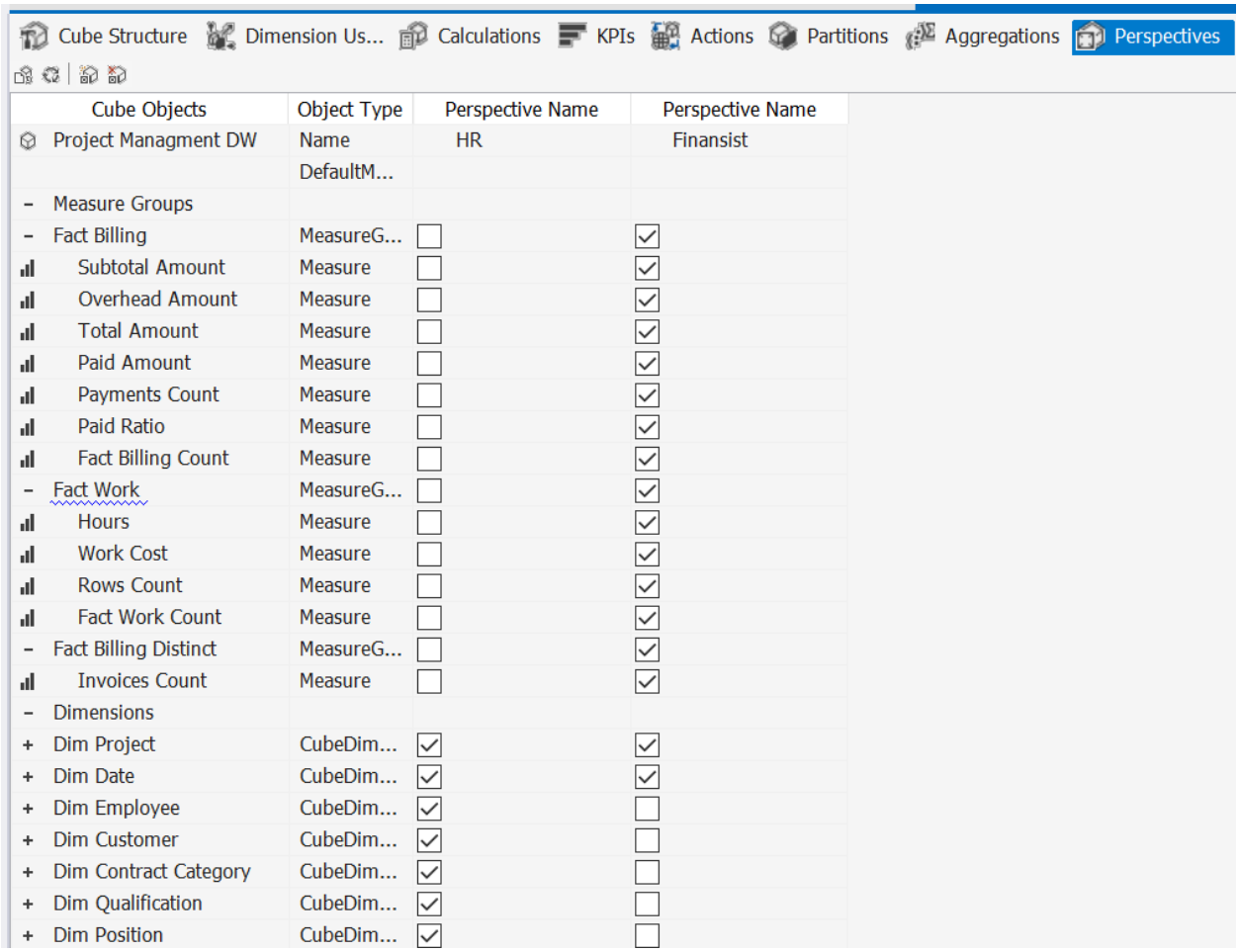


Рисунок 19. Створення Perspectives

4.7. Розгортання та тестування куба

- **Розгортання (Deployment):** Проект успішно розгорнуто на локальному сервері Analysis Services.
- **Обробка (Processing):** Виконано повну обробку куба (Process Full). Логи підтвердили успішне зчитування всіх рядків з Data Warehouse та побудову індексів.
- **Тестування:** За допомогою вбудованого браузера куба та підключення через Excel перевірено коректність цифр (наприклад, чи сходиться сума зарплати по всіх відділах із загальним підсумком). Час виконання складних аналітичних запитів складає частки секунди, що підтверджує ефективність оптимізації.
- **Резервне копіювання:** Створено файл резервної копії (.abf) бази даних SSAS для забезпечення відновлюваності.

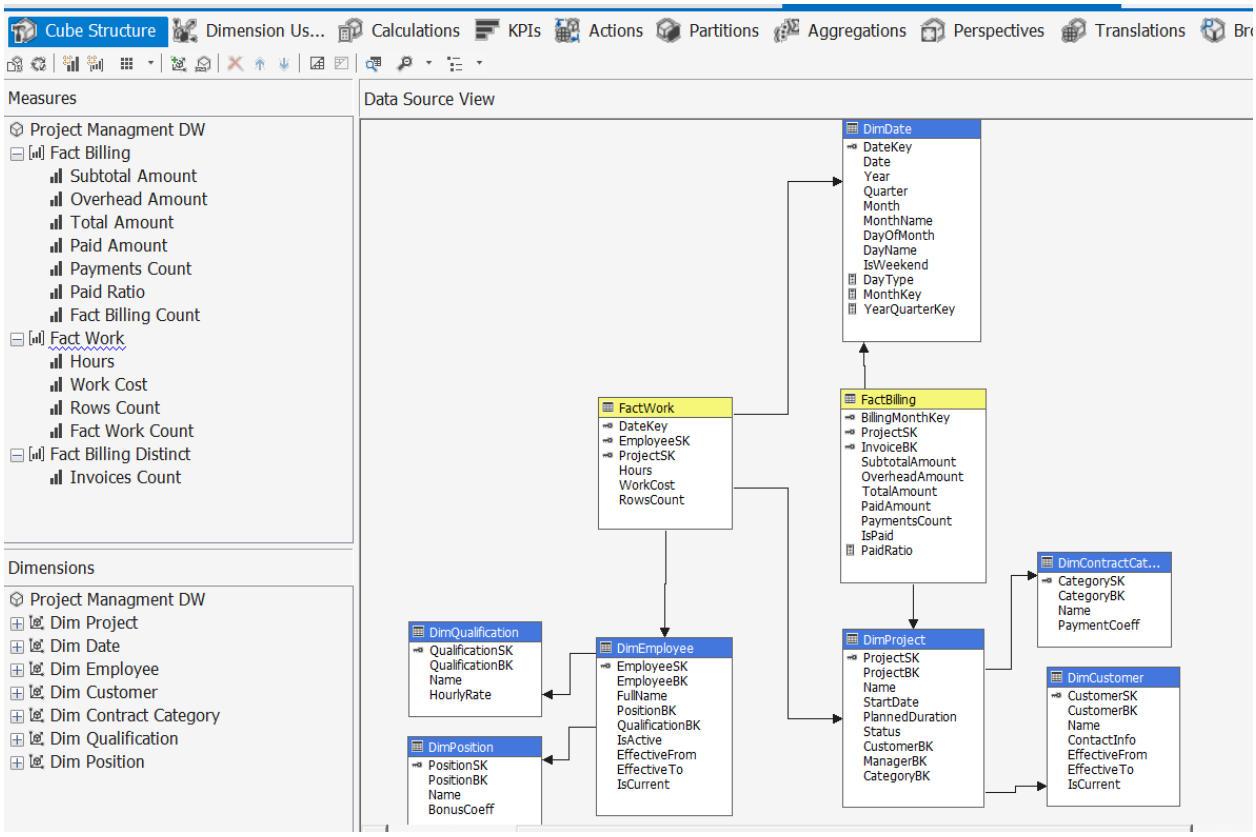


Рисунок 20. OLAP Cube

Розділ 5. Створення аналітичних звітів (SQL Server Reporting Services)

5.1. Створення проекту SSRS та налаштування середовища

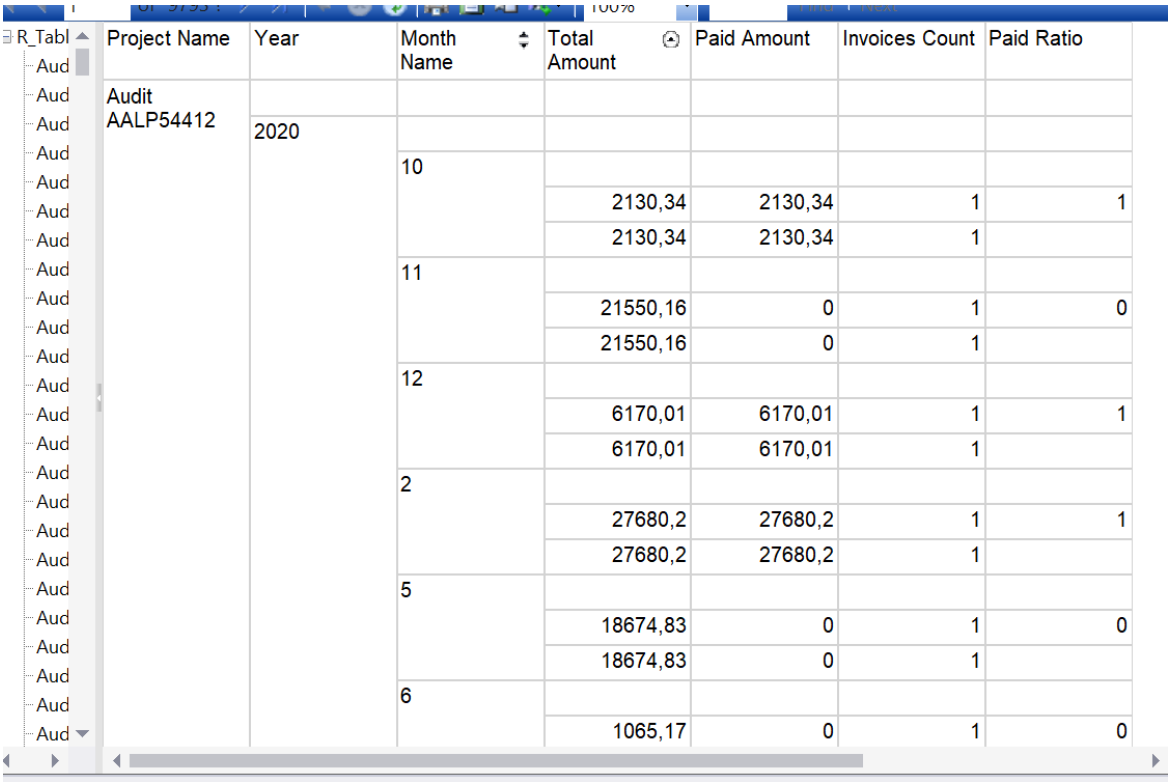
Для візуалізації даних та створення регламентованої звітності використано платформу SQL Server Reporting Services (SSRS). Розробка виконувалася у середовищі SQL Server Data Tools (SSDT). На початковому етапі створено новий проект типу Report Server Project. Ключовим кроком стало налаштування джерела даних. Замість прямого підключення до реляційної бази (що могло б сповільнити її роботу), було створено Shared Data Source (Спільне джерело даних), яке підключається безпосередньо до OLAP-куба Analysis Services, розробленого на попередньому етапі. Це забезпечує:

1. Миттєвий доступ до попередньо розрахованих агрегацій.
2. Єдину точку входу для всіх звітів (зміна пароля чи сервера відбувається в одному місці).
3. Можливість використання ієрархій та KPI, закладених у кубі.

5.2. Розробка звітів

У рамках курсової роботи реалізовано комплекс із 5 звітів, що покривають потреби різних рівнів менеджменту. Для побудови запитів до куба використовувався графічний конструктор MDX-запитів.

Табличний звіт (Table Report)



Project Name	Year	Month Name	Total Amount	Paid Amount	Invoices Count	Paid Ratio
Audit AALP54412						
	2020					
		10				
			2130,34	2130,34	1	1
			2130,34	2130,34	1	
		11				
			21550,16	0	1	0
			21550,16	0	1	
		12				
			6170,01	6170,01	1	1
			6170,01	6170,01	1	
		2				
			27680,2	27680,2	1	1
			27680,2	27680,2	1	
		5				
			18674,83	0	1	0
			18674,83	0	1	
		6				
			1065,17	0	1	0

Рисунок 21. Table Report

Для реалізації типу цього звіту було створено датасет з полями ProjectName, Year, MonthName, Total Amount, Paid Amount, Invoices Count, Paid Ratio.

Матричний звіт (Matrix Report)

	[Year]					
Project Name	[Month_Name]			Total		
[ProjectName]	[Sum(Paid_Am	[Sum(Paid_Rat	[Sum(Total_Am	[Sum(Paid_Am	[Sum(Paid_Rat	[Sum(Total_Am
Total	[Sum(Paid_Am	[Sum(Paid_Rat	[Sum(Total_Am	[Sum(Paid_Am	[Sum(Paid_Rat	[Sum(Total_Am

Рисунок 22. Matrix Report 1

Audit CFBU 05191	0	0	3048,07				
Audit CGGD21456				0	0	7615,61	
Audit CGON 80934	4921,2	1	4921,2	3097,12	1	3097,12	
Audit CHKC 64698	766,81	1	766,81	0	0	13353,83	4855,
Audit CIHV 25444				0	0	2276,96	
Audit CLKZ07006				10685,5	1	10685,5	3353,
Audit CMVS53423	0	0	8090,13	420,07	1	420,07	5887
Audit CNRM37361	0	0	30268,59	5858,02	1	5858,02	12503
Audit CNUJ 19530	0	0	5587,62	0	0	21589,43	7324,
Audit CPNE 05039	0	0	4931,12				
Audit CQRE 12277	10450,36	1	10450,36	869,74	1	869,74	
Audit CRNE 62795	0	0	13190,13	26292,98	1	26292,98	

Рисунок 23. Matrix Report 2

Цей звіт відображає розхід фінансів щомісяця щороку. Також було створено окремий датасет з полями ProjectName, Year, MonthName, Total Amount, Paid Amount, Invoices Count, Paid Ratio.

Звіти з діаграмами (Chart Reports)

Рисунок 24. Line and Column charts

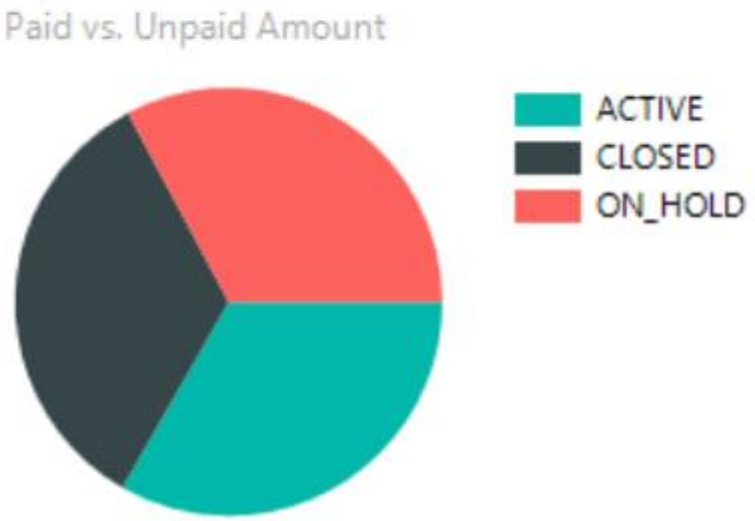


Рисунок 25. Pie chart

5.3. Додаткова функціональність звітів

Для підвищення зручності користування (Usability) та інформативності звітів впроваджено ряд інтерактивних функцій.

1. **Умовне форматування (Conditional Formatting):** У фінансовому звіті застосовано кольорову індикацію для поля «Дебіторська заборгованість».
    - *Умова:* Якщо сума боргу > 0 і прострочення > 30 днів — фон комірки стає **червоним**.

- *Умова:* Якщо рахунок оплачено повністю — фон **зелений**. Це дозволяє бухгалтеру миттєво фокусувати увагу на проблемних зонах.

2. **Функціонал експорту (Export):** Звіти оптимізовано для експорту в популярні формати. Перевірено коректність відображення кирилиці та меж таблиць при вивантаженні у **PDF** (для друку) та **Excel** (для подальшої обробки формул користувачем).
3. **Вкладені звіти (Subreports):** Реалізовано механізм деталізації (Drill-through). У головному звіті «Список проектів» назва проекту є гіперпосиланням. При натисканні на неї відкривається окремий детальний звіт «Картка проекту», куди параметром передається ProjectID.
4. **Розбивка на сторінки (Page Breaks):** У звіті «Рахунки для клієнтів» налаштовано примусовий розрив сторінки після кожної групи (Замовника). Це дозволяє масово роздрукувати рахунки, де кожен клієнт буде на окремому аркуші.

Висновки:

У ході виконання курсової роботи було успішно вирішено завдання проектування та розробки комплексної інформаційно-аналітичної системи для предметної області «Виконання проектів». Актуальність проведеного дослідження зумовлена необхідністю ефективної обробки великих масивів даних, які накопичуються проектно-орієнтованими підприємствами в процесі їхньої діяльності, та критичною важливістю цих даних для прийняття зважених управлінських рішень.

На початковому етапі було здійснено системний аналіз бізнес-процесів, пов’язаних із плануванням ресурсів, фіксацією трудовитрат та фінансовими розрахунками із замовниками. Це дозволило спроектувати реляційну модель бази даних, яка відповідає третій нормальній формі та забезпечує надійне зберігання інформації. Для верифікації спроектованої системи було розроблено стратегію генерації даних, в результаті реалізації якої база даних була наповнена синтетичним, але реалістичним масивом інформації обсягом понад 500 000 записів, що охоплюють п’ятирічний період діяльності компанії. Це дозволило імітувати реальне навантаження та підтвердити масштабованість розробленої архітектури.

Ключовим етапом роботи стала реалізація процесів інтеграції даних (ETL) засобами SQL Server Integration Services. Було розроблено пакети для автоматизованого переміщення даних з операційної бази в аналітичне сховище (Data Warehouse), побудоване за схемою «зірка». Впровадження механізмів очищення даних, трансформації ключів та обробки повільно змінюваних вимірів (SCD Type 2) забезпечило історичну достовірність та узгодженість інформації для подальшого аналізу.

На базі створеного сховища було розроблено багатовимірний OLAP-куб у середовищі SQL Server Analysis Services. Використання технології багатовимірного аналізу дозволило реалізувати складні бізнес-метрики, такі як аналіз маржинальності проектів, розрахунок ефективності співробітників та моніторинг динаміки доходів (YoY) за допомогою мови MDX. Завдяки налаштованим агрегаціям та партиціюванню вдалося досягти високої швидкодії виконання аналітичних запитів.

Практичну цінність роботи підтверджено створенням набору інтерактивних звітів на платформі SQL Server Reporting Services. Таким чином, мета курсової роботи досягнута в повному обсязі: створено цілісну систему Business Intelligence, яка дозволяє трансформувати накопичені транзакційні дані у якісну аналітику для стратегічного управління підприємством.

Список використаних джерел:

1. Microsoft. SQL Server Documentation. — Available at: https://learn.microsoft.com/sql
2. Microsoft. SQL Server Integration Services (SSIS) Documentation. — Available at: https://learn.microsoft.com/sql/integration-services
3. Microsoft. SQL Server Analysis Services (SSAS) Documentation. — Available at: https://learn.microsoft.com/sql/analysis-services
4. Microsoft. Data Warehouse Design Concepts. — Available at: https://learn.microsoft.com/azure/architecture/data-guide/relational-data/data-warehousing
5. Redgate Software. SQL Data Generator Documentation. — Available at: https://www.red-gate.com/products/sql-development/sql-data-generator