

Вступ до баз даних

Introduction to databases

Тема 1

Торіс 1

1.1 Бази даних та СУБД

1.1 Databases and DBMS

- Поява у 1960-х запам'ятовуючих пристроїв відносно великої ємності відкрило широкі можливості для створення складних структур довгострокового зберігання даних.
- Appearance in 1960-s memory devices of relatively big capacity allowed wide opportunities for development of complex structures of long-term data storage.

1.1 Бази даних та СУБД

1.1 Databases and DBMS

- Ці можливості призвели до значного ускладнення коду програм, подорожчання їх розробки та зниженню надійності. З'явилась ідея централізації функцій управління даними та створення систем, що надають програмам послуги з обробки даних – систем управління базами даних (СУБД).
- These possibilities led to increase in complexity of application code, development cost, and decrease of reliability. Appeared an idea of centralization of data management functions and creation of systems that provide data processing services to applications – database management systems (DBMS).

1.1 Бази даних та СУБД

1.1 Databases and DBMS

- СУБД забезпечують високу надійність зберігання та ефективність обробки даних, недосяжні при розробці індивідуальних засобів управління даними для кожної програми.
- DBMS provide high reliability of data storage and high efficiency of data processing, that are not possible with development of individual data management tools for each application.

1.1 Бази даних та СУБД

1.1 Databases and DBMS

- Особливості ранніх СУБД були обумовлені вимогами програм оперативної обробки даних (online transaction processing, OLTP) у банківській та фінансовій сферах.
- Features of early DBMS were defined by requirements of online transaction processing (OLTP) applications in banking and financial industries.

1.1 Бази даних та СУБД

1.1 Databases and DBMS

- СУБД – програмний комплекс, що забезпечує централізоване зберігання даних та надає програмам послуги з обробки даних.
- DBMS – software system that ensures centralized data storage and provides data processing services to applications.

1.1 Бази даних та СУБД

1.1 Databases and DBMS

- База даних (БД) – сукупність даних, що зберігаються під управлінням СУБД. БД – фундамент, на якому будуються майже будь-які програми. СУБД, пов’язана з деякою конкретною БД та готова виконувати запити на обробку цієї БД, називається екземпляром (instance) або сервером БД.
- Database (DB) – dataset stored under DBMS management. DB – baseline of almost any applications. DBMS, which is related to some specific DB and ready to run DB processing queries, is called instance or DB server.

1.2 Вимоги до СУБД

1.2 Requirements to DBMS

- Ранні СУБД дуже сильно відрізнялися своєю внутрішньою організацією та можливостями. Знадобилось декілька років, для того щоб визначити основні функції СУБД та вимоги, які слід ставити до таких систем.
- Early DBMS were very different by their internal organization and features. It took several years in order to define main DBMS functions and requirement for such systems.

1.2 Вимоги до СУБД

1.2 Requirements to DBMS

- Системи OLTP характеризуються тим, що:
 - Кожна операція займає дуже мало часу (мілісекунди)
 - Дані спільно використовуються багатьма програмами
 - Операції використовують незначну долю загального обсягу даних
- OLTP systems are characterized by the following features:
 - Each operation takes very small time (milliseconds)
 - Data used together by multiple applications
 - Operations are using insignificant piece of total amount of data

1.2 Вимоги до СУБД

1.2 Requirements to DBMS

- Процеси обробки та структури даних в областях, де використовувались ранні СУБД, були формалізовані задовго до появи обчислювальних систем (наприклад, бухгалтерський облік). Це привело до того, що СУБД, як правило, орієнтовані на обробку структурованих даних.
- Data processing workflows and data structures, where early DBMS were used, were formalized long before occurrence of computing systems (e.g., accounting). It led to the fact that DBMS are mostly oriented on structured data processing.

1.2 Вимоги до СУБД

1.2 Requirements to DBMS

Основні вимоги до СУБД:

- Розділення програм і даних. Опис структури даних відділений від коду програм. Система повинна допускати незалежну зміну структури даних та коду програми.
- Високорівнева мова запитів. Система повинна надавати засоби обробки даних, незалежні від якоїсь програми.
- Цілісність. Система повинна запобігати запису даних, що порушують заздалегідь визначені обмеження.
- Узгодженість. Система повинна запобігати пошкодженню даних внаслідок паралельної роботи декількох програм.
- Відмовостійкість. СУБД не повинна допускати втрати даних.
- Захист та розмежування доступу. Система повинна запобігати несанкціонованому доступу до даних та надавати кожному користувачу доступ до даних згідно з його правами.

1.2 Вимоги до СУБД

1.2 Requirements to DBMS

General requirements to DBMS:

- Independence of applications and data. Description of data structure is separated from applications code. System should allow independent change of data structure and applications code.
- High-level query language. System should provide data processing tools, independent from any application.
- Integrity. System should not allow data insertion that violate previously defined constraints.
- Consistency. System should not allow data damage as the result of concurrent work of multiple applications.
- Fault tolerance. DBMS should not allow data loss.
- Security and access control. System should not allow unauthorized access to data and should allow access to data to each user according to its privileges.

1.2 Вимоги до СУБД

1.2 Requirements to DBMS

- У ранні роки існування СУБД передбачалось, що дані інформаційних систем підприємства будуть зберігатися у єдиній БД. На практиці це ніколи не було реалізовано: зазвичай для кожної програми або групи програм створюється окрема БД.
- In early years of DBMS was assumed that data of information systems of organization will be stored in a single DB. In practice this was never implemented: usually for each application or group of applications a separate DB is created.

1.3 Розділення даних та програм

1.3 Independence of data and applications

- Кожна СУБД створює деякий рівень абстракції для інших програмних компонент, які використовують її послуги. Необхідно щоб опис структури даних був загальний для усіх програм, що призводить до ідеї відділення опису структури даних від програм. Такий опис зберігається у самій БД і називається схемою бази даних.
- Each DBMS creates some level of abstraction for other software components that use its services. It is required to data structure description is general for all applications, which leads to the idea of separation of data structure from applications. Such description is stored in DB itself and it is called database scheme.

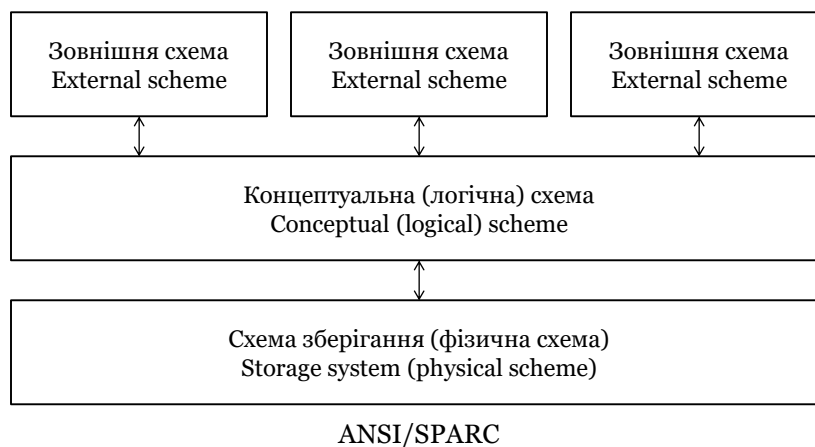
1.3 Розділення даних та програм

1.3 Independence of data and applications

- Для визначення схем використовуються мови опису даних. Для реалізації ідеї відокремлення даних від програм було запропоновано модель мови опису даних ANSI/SPARC.
- To define schemes data description languages are used. In order to implement the idea of independence of data and applications the data description language models ANSI/SPARC was proposed.

1.3 Розділення даних та програм

1.3 Independence of data and applications



1.3 Розділення даних та програм

1.3 Independence of data and applications

Модель ANSI/SPARC включає:

- Зовнішню схему, що містить опис даних у вигляді, у якому вони будуть використовуватися програмами, а також відображення логічної структури даних у зовнішню схему.
- Концептуальну схему, що містить повний опис логічної структури даних, доступний для СУБД (логічна схема БД).
- Схему зберігання, що описує яким чином організоване зберігання логічних структур даних (фізична схема БД).

1.3 Розділення даних та програм

1.3 Independence of data and applications

ANSI/SPARC model includes:

- External scheme, which contains data description in the form in which they will be used by applications, as well as the mapping of logical data structure into external scheme.
- Conceptual scheme, which contains full description of logical data structure, accessible by DBMS (logical DB scheme).
- Storage scheme, which describes how storage of logical data structures is organized (physical DB scheme).

1.3 Розділення даних та програм

1.3 Independence of data and applications

- В ідеалі така модель забезпечує еволюцію програм та системи. При появі нових програм достатньо визначити нову зовнішню схему. В результаті впровадження нової програми не вплине на роботу інших програм. Нові структури, додані до концептуальної схеми, не вплинуть на роботу інших програм, оскільки їх зовнішні схеми не будуть містити нових елементів даних.
- Ideally such model provides evolution of applications and a system. When new programs appeared it is sufficient to define new external scheme. As the result, implementation of new application will not affect other applications. New structures, added to the conceptual scheme, will not affect other applications, since their external schemes do not contain new data elements.

1.3 Розділення даних та програм

1.3 Independence of data and applications

- У реальності еволюційні можливості можуть бути реалізовані лише у випадку, коли розробники БД та програм їх ретельно враховують. Еволюція БД має сенс лише коли цінність накопичених даних є високою. У інших випадках створення нової БД є більш виправданим рішенням.
- In reality evolution capabilities could be implemented only in case if DB and applications developers carefully follow these capabilities. DB evolution is reasonable only in case of stored data is highly valuable. In other cases development of new DB is more reasonable solution.

1.3 Розділення даних та програм

1.3 Independence of data and applications

- У останні роки багато методологій розробки програм передбачають генерацію схеми БД на основі об'єктної моделі програми для швидкої розробки прототипів. При цьому не зникає складність та необхідність вирішувати задачі проектування БД.
- Recently many software development methodologies assume generation of DB scheme based on application object model for rapid prototypes developments. At the same time complexity is not reduced and it is still required to solve DB design problems.

1.3 Розділення даних та програм

1.3 Independence of data and applications

- Особливості проектування схеми БД суттєво залежать від моделі даних, що застосовується. Складність проектування може по-різному розподілятися між БД та програмою: чим бідніше модель даних, тим більше необхідно зробити на рівні програми.
- Features of DB scheme design depend on data model, which is used. Design complexity is differently distributed between DB and application: if the data model is poor, the more work need to be done on the application layer.

1.4 Мови запитів

1.4 Query languages

- Наявність опису логічної структури даних дозволяє виконувати достатньо складні операції маніпулювання даними у СУБД. Такі операції записуються на мові запитів. Мови сучасних СУБД є декларативними – дозволяють визначити необхідний результат, але не спосіб виконання запиту. СУБД обирає найбільш ефективні (за деяким критерієм) алгоритми отримання результату.
- Existence of logical data structure allows to execute complex enough operations of data manipulation in DBMS. These operations are written in a query language. Languages of modern DBMS are declarative – allow to define required result, but not the way to execute query. DBMS chooses the most effective (by some criteria) algorithms to obtain the result.

1.4 Мови запитів

1.4 Query languages

- Практика розробки програм без використання мов запитів призвела до появи так званих NoSQL систем. При використанні таких систем частина функцій СУБД переноситься у програму, що призводить до підвищення складності програми та вартості її розробки або до зниження якості, що у деяких випадках є припустимим.
- Practice of application development without using query languages led to occurrence of so-called NoSQL systems. When using these systems some DBMS functions are transferred to application, which led to increasing of applications complexity and development cost, or to quality decreasing, which is acceptable in some cases.

1.5 Цілісність та узгодженість

1.5 Integrity and consistency

- Окрім опису структур даних до логічної схеми БД можуть бути включені додаткові умови – обмеження цілісності. СУБД перевіряє обмеження цілісності при виконанні будь-яких змін даних та не допускає порушень цих обмежень. Це дозволяє значно спростити розробку програм та підвищити їх якість.
- Besides data structured descriptions, DB scheme can contain additional conditions – integrity constraints. DBMS checks integrity constraints when executing any data changes and does not allow violations of these constraints. This allows significantly simplify applications development and increase their quality.

1.5 Цілісність та узгодженість

1.5 Integrity and consistency

- Стан БД, у якому виконуються умови предметної області, називається узгодженим. Кінцевий набір операцій, що переводить БД з одного узгодженого стану до іншого, називається транзакцією. Однією з важливих функцій СУБД є запобігання порушенням узгодженості при одночасній роботі декількох програм (або користувачів).
- DB state in which domain conditions are satisfied is called consistent state. Final set of operations that transfer DB from one to another consistent state is called transaction. One of the main DBMS functions is prevention of consistency violations when concurrent applications (or users) are working.

1.6 Відмовостійкість

1.6 Fault tolerance

- Відмови систем призводять до зупинки бізнес-процесів, а втрати даних призводять до катастрофічних подій (не тільки для функцій підприємства, а і для життя людей та стану навколишнього середовища). Сучасні СУБД можуть гарантувати повну збереженість даних та відновлення після відмови в узгодженому стані за долі секунди.
- System faults lead to stopping of business processes, while data losses lead to catastrophic events (not only for enterprise functions, but also for human lives and environment condition). Modern DBMS can guarantee full safety of data and recovery in consistent state in milliseconds after faults.

1.6 Відмовостійкість

1.6 Fault tolerance

- Висока вартість пов'язана з необхідністю багаторазового дублювання засобів на усіх рівнях, починаючи з обладнання. При використанні хмарних сервісів необхідно мати додаткові ресурси, здатні забезпечити виживання при відмові. Тому при проектуванні системи необхідно обирати рівень відмовостійкості, що є дійсно необхідним.
- High cost is related to the requirement of presence of multiple duplicates on all layers, starting from hardware. When using cloud services, it is required to have additional resources, that could provide survival in case of faults. Therefore, when designing system it is necessary to choose really required level of fault tolerance.

1.7 Безпека та обмеження доступу

1.7 Security and access control

- Дані потребують захисту не тільки від відмов, а й від несанкціонованого доступу. Усі сучасні СУБД надають засоби перешкоджання та обмеження доступу. В деяких системах засоби захисту СУБД взагалі не використовуються, але по-справжньому надійний захист повинен бути багаторівневим, а деякі види захисту неможливо реалізувати без СУБД.
- Data should be protected not only from faults, but also from unauthorized access. All modern DBMS provide access prevention and control tools. In some systems security features of DBMS are not used at all, but really reliable protection should be multi-layered, while some protection types are not possible without DBMS.

1.8 Продуктивність

1.8 Performance

- Найбільш важливими метриками є пропускна здатність та час відгуку системи. Для СУБД це запити або інші дії різної складності. Пропускна здатність – середня кількість подібних дій за одиницю часу. Час відгуку – середній час виконання дії. Покращення однієї з характеристик не обов'язково призведе до покращення іншої.
- The most important metrics are throughput and response time. For DBMS there are queries or other actions of different complexity. Throughput is an average number of similar actions at a time. Response time is an average time of execution. Improvement of one of these metrics will not necessary lead to improvement of another.

1.8 Продуктивність

1.8 Performance

- Важливо розрізняти вимір на стороні клієнта або сервера. Час виконання на сервері може бути меншим за час відправлення запиту та повернення відповіді. Тому для веб-застосунків найбільш важливим є час генерації HTML сторінки у відповідь на запит користувача.
- It is important to differentiate between the measure on client or server side. Execution time on server could be less than time required to send request and receive response. Then for web-applications generation time of HTML pages in response to users request is more important.

1.8 Продуктивність

1.8 Performance

- Для паралельних систем найбільш важливою характеристикою є масштабованість (пропускної здатності або часу відгуку). Необхідно оцінити характеристику з певним обсягом даних, навантаженням та одним обчислювачем, а також ту ж саму характеристику з обсягом даних, навантаженням та обчислювачами, збільшеними у N разів.
- For concurrent systems the most important metric is scalability (of throughput or response time). It is required to measure some metric with some data amount, workload, and one computation instance, and then the same metric with data amount, workload, and computation instances increased N times.

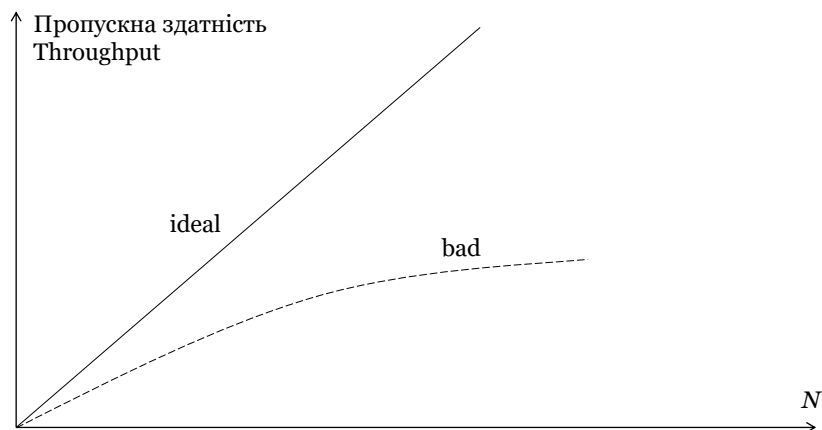
1.8 Продуктивність

1.8 Performance

- Ідеальна масштабованість за пропускнуою здатністю може бути представлена лінійною залежністю: система, що містить у N разів більше обладнання і даних може обробляти у N разів більше запитів. В реальності така масштабованість недосяжна через витрату ресурсів на синхронізацію паралельних обчислень.
- Ideal scalability of throughput could be described using linear dependency: system that contains N times more hardware and data could process N times more queries. In reality such scalability is not reachable because of resources usage to synchronize parallel computations.

1.8 Продуктивність

1.8 Performance



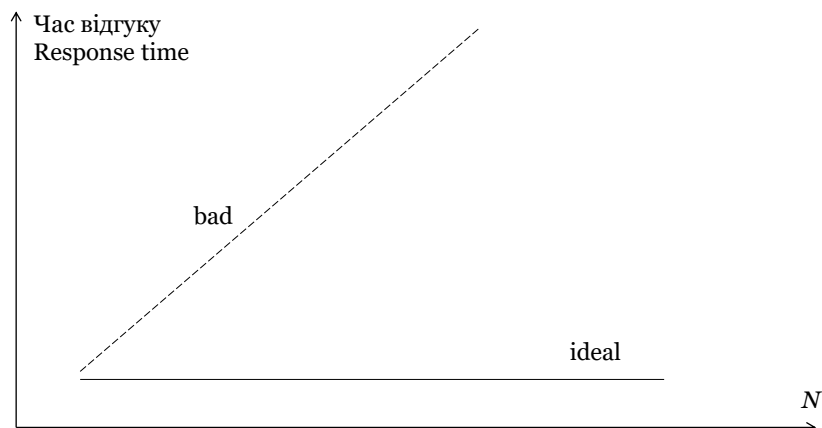
1.8 Продуктивність

1.8 Performance

- Для масштабованості за часом відгуку ідеальна залежність представляється константою: при збільшенні кількості обчислювачів, обсягу даних та потоку запитів час відгуку не зростає. Ідеальна масштабованість за часом відгуку практично недосяжна.
- For scalability of response time ideal dependency is described using the constant: when increasing number of hardware, data amount, and queries flow response time is not increasing. Ideal scalability of response time is almost not reachable.

1.8 Продуктивність

1.8 Performance



1.8 Продуктивність

1.8 Performance

- Прискорення = час відгуку на системі з одним обчислювачам / час відгуку на системі з N обчислювачами.
Доступність = час нормальної роботи системи / інтервал часу виміру доступності. Усі інтегральні метрики корисні для оцінки роботи системи в цілому, але не для окремих операцій або запитів програм.
- Acceleration = response time for system with one computing instance / response time for system with N computing instances.
Availability = time of normal work of a system / time range used to measure availability. All general metrics are useful to estimate system work in general but not for a single operations or applications queries.

1.9 Створення програм, працюючих з БД

1.9 Development of applications to work with DB

- Сучасні СУБД поставляються з інструментами адміністрування та підтримки БД (створення та зміна таблиць, редагування записів, управління доступом та резервними копіями). Але такий інтерфейс не підходить для бізнес-застосувань. Система повинна бути зрозумілою для масового користувача, який без усякого навчання міг би почати з нею працювати.
- Modern DBMS are provided with administrating and maintenance tools for DB (creating and editing tables, editing records, managing access and backups). But such interface does not suite for business applications. Systems should be understandable for mass user that could start work with it without any training.

1.9 Створення програм, працюючих з БД

1.9 Development of applications to work with DB

- Для створення бізнес-застосунків використовується клієнт-серверна архітектура. СУБД працює на сервері, а програма – на клієнті. Бізнес-логіка може бути реалізована як на сервері у вигляді збережених процедур, так і на клієнті за допомогою мови програмування.
- Client-server architecture is used to create business applications. DBMS works on the server, while the application works on the client side. Business logic could be implemented on the server in the form of stored procedures, as well as on the client using the programming language.

1.9 Створення програм, працюючих з БД

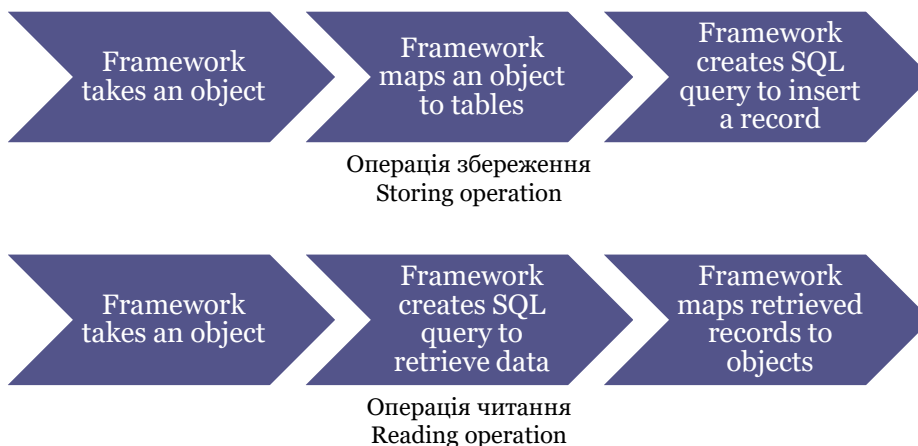
1.9 Development of applications to work with DB

- Проблема невідповідності об'єктно-орієнтованих мов програмування та реляційних баз даних вирішуються за допомогою фреймворків об'єктно-реляційного відображення. Розробник працює зі звичними об'єктними моделями, які автоматично перетворюються у таблиці та навпаки.
- Problem of mismatch between object-oriented programming languages and relational databases are solved using the object-relational mapping (ORM) frameworks. Developer works with usual object models, that automatically transformed into tables and vice versa.

41

1.9 Створення програм, працюючих з БД

1.9 Development of applications to work with DB



42

1.9 Створення програм, працюючих з БД

1.9 Development of applications to work with DB

- Використання фреймворків прискорює розробку, оскільки програмісту не потрібно глибоко знати SQL або реляційну теорію. Однак фреймворки не дозволяють тонкого налаштування запитів. Хорошим рішенням є використання фреймворків для стандартних операцій та чистого SQL для складних запитів.
- Using frameworks accelerates development, since developer does not need to have deep knowledge of SQL or relational theory. But frameworks do not allow tuning of queries. Good practice is to use frameworks for standard operations and pure SQL for complex queries.

1.9 Створення програм, працюючих з БД

1.9 Development of applications to work with DB

- Спроба вирішити проблему невідповідності призвела до створення баз даних NoSQL, які представляють собою альтернативу реляційним СУБД. Такі БД не мають структурованої схеми та працюють напряду з об'єктами.
- An attempt to solve the mismatch problem led to occurrence of NoSQL databases, which are alternatives to relational DBMS. Such DB do not have structured scheme and operate directly to objects.

Контрольні питання

1. Основні вимоги до СУБД.
2. Основні компоненти моделі ANSI/SPARC.
3. Основні особливості мов запитів у сучасних СУБД.
4. Поняття незалежності даних.
5. Переваги використання незалежності даних.
6. Поняття обмежень цілісності та узгодженості даних у СУБД.
7. Поняття безпеки та розмежування доступу у сучасних СУБД.
8. Основні метрики оцінки продуктивності.
9. Архітектура клієнт-сервер.
10. Об'єктно-реляційна втрата відповідності.

Assessment questions

1. Main requirements to DBMS.
2. Main components of ANSI/SPARC model.
3. Main features of query languages in modern DBMS.
4. Data independence.
5. Advantages of data independence.
6. Integrity constraints and data consistency in DBMS.
7. Security and access control in modern DBMS.
8. Main performance metrics.
9. Client-server architecture.
10. Object-relational mismatch.

Зберігання даних та файлова структура Data storage and file structure

Тема 2

Topic 2

2.1 Класи застосувань БД

2.1 DB applications classes

- Однією з передумов створення СУБД стала поява пристроїв зберігання даних з довільним доступом та відносно великої ємності. На сьогоднішній день існує 2 класи задач використання СУБД:
 - OLTP (Online Transaction Processing) – системи оперативної обробки даних;
 - OLAP (Online Analytical Processing) – системи аналітичної обробки (великих) даних.
- One of prerequisites for DBMS appearance was occurrence of data storage devices of random access and relatively high capacity. Today there are 2 classes of DBMS use cases:
 - OLTP (Online Transaction Processing) – operative data processing systems;
 - OLAP (Online Analytical Processing) – analytical (big) data processing systems.

2.1 Класи застосувань БД

2.1 DB applications classes

- Системи OLTP характеризуються відносно великим потоком коротких транзакцій, що обробляють дуже невелику кількість записів БД. До цього класу належать сучасні веб-застосування, у яких в ролі транзакцій виступають HTTP-запити.
- OLTP systems are characterized by relatively large stream of short transactions, which processing very small number of DB records. This class includes modern web-applications, in which transactions are implemented as HTTP requests.

2.1 Класи застосувань БД

2.1 DB applications classes

- Системи OLAP призначені для формування узагальнених звітів, що отримуються в результаті обробки усіх або значної долі записів у БД. У таких системах повністю відсутнє оновлення даних, що робить вимоги цілісності та узгодженості БД менш актуальними.
- OLAP systems are aimed to form generalized reports created as the result of processing of all DB records or major set of DB records. In such systems update operations are not executed at all, which makes DB integrity and consistency requirements less relevant.

2.1 Класи застосувань БД

2.1 DB applications classes

- Особливий клас – системи спільного редагування та розробки. В таких системах використовуються об'єктно-орієнтовані СУБД або спеціалізовані надбудови над файловими системами. У сучасні СУБД інтегровано функціонал для зберігання та обробки документів у слабкоструктурованих форматах. Існують особливі системи обробки дуже великих послідовностей або дуже великих графів.
- Special class includes cooperative editing and development systems. Such systems use object oriented DBMS or specialized customizations of file systems. Modern DBMS include functions to store and process documents of weakly structured formats. There are special systems for large sequences processing or very large graphs processing.

2.2 Структури зберігання

2.2 Storage structures

- Об'єкти у OLTP системах характерні невеликою кількістю атрибутів, майже усі з яких використовуються у кожному запиті. Тому системи OLTP розміщують атрибути кожного об'єкту у суміжних ділянках пам'яті та групують пов'язані об'єкти, що обробляються разом. Такий спосіб називається “зберігання у рядках”.
- Object in OLTP systems are characterized by small number of attributes, almost all of them are used in each query. Then OLTP systems allocate attributes of each object in neighbor memory areas and group related objects that processed together. Such way is called “row oriented” storage.

2.2 Структури зберігання

2.2 Storage structures

- Для задач OLAP характерні записи, що містять велику кількість атрибутів, однак у кожному запиті використовується їх невелика підмножина. Тому альтернативний спосіб “зберігання у колонках” є більш ефективним через швидке послідовне сканування, можливість стиснення даних, відсутність оновлень.
- OLAP problems are characterized by records with large number of attributes, but in each query there are used only small subset of attributes. Then alternative “column oriented” storage is more effective because of fast full table scan, data compression possibility, absence of updates.

2.3 Архітектури зв'язку з програмами

2.3 Application communication architectures

- Однокористувацькі СУБД. Відсутня можливість спільного використання даних, транзакцій тощо. Замість клієнт-серверної архітектури файли БД розміщуються на файл-сервері. Зараз однокористувацькі СУБД використовуються в ролі вбудованих систем у мобільних пристроях.
- Single-user DBMS. Shared data usage, transactions, etc. are not supported. Instead of client-server architecture DB files are placed on the file-server. Now single-user DBMS are used as embedded systems in mobile devices.

2.3 Архітектури зв'язку з програмами

2.3 Application communication architectures

- СУБД, що використовують архітектуру клієнт-сервер, виявились недостатньо масштабованими за кількістю з'єднань для OLTP систем, що характеризуються дуже великою кількістю користувачів. Рішенням даної проблеми стала багатоварова архітектура.
- DBMS that use client-server architecture are not enough scalable by number of connections for OLTP systems, which have very big number of users. Multi-tier architecture became the solution of this problem.

2.3 Архітектури зв'язку з програмами

2.3 Application communication architectures

- Багатошарова архітектура. Одна частина програми виконується на клієнті, а інша частина – на сервері застосувань. Програма також є клієнтом сервера БД та одночасно – сервером, що обслуговує наступний рівень клієнтів (наприклад, виконує HTTP-запити).
- Multi-tier architecture. One part of application is executed on the client, while another part is executed on the application server. Application is still DB server client but at the same time is a server that supports the next layer of clients (e.g., executes HTTP requests).

2.3 Архітектури зв'язку з програмами

2.3 Application communication architectures

- Сервер застосувань не зберігає спільних даних, тому з'являється можливість горизонтального масштабування. Але багаторазове використання сеансів роботи з БД призводить до накопичення тимчасових об'єктів. Також додаткового узгодження потребує кешування даних на різних серверах застосувань.
- Application server does not store shared data, so the capability of horizontal scaling becomes possible. However, multiple usage of DB sessions leads to accumulation of multiple temporary objects. Also caching of data on multiple application servers require additional control.

2.4 Обладнання. Носії даних

2.4 Hardware. Storage devices

- Традиційно для зберігання даних використовуються магнітні диски. А обробка фактично відбувається в буфері у оперативній пам'яті. Швидке зростання обсягів та зниження вартості оперативної пам'яті викликав зростання інтересу до СУБД, що зберігають дані у оперативній пам'яті.
- Traditionally magnetic storage drives were used to store data. While data processing is in fact is performed in the cache of random access memory. Fast growth of volumes and decrease of cost of the random access memory (RAM) has led to increasing of interest for DBMS that store data in the RAM.

2.4 Обладнання. Носії даних

2.4 Hardware. Storage devices

Основні підходи організації БД у оперативній пам'яті:

- Традиційні СУБД з дуже великим обсягом оперативної пам'яті, виділеної для кешування з метою підвищення продуктивності
- Системи копіювання даних в пам'ять на рівні логічної схеми (працюють разом з традиційними СУБД), а не на рівні структур зберігання даних
- Розподілені системи БД у оперативній пам'яті, в яких кожний елемент даних зберігається у декількох копіях на різних вузлах системи (оскільки відмова усіх вузлів малоімовірна)
- Системи зберігання даних виключно для читання (БД для аналітичної обробки даних)

2.4 Обладнання. Носії даних

2.4 Hardware. Storage devices

Main approaches to DB organization in RAM:

- Traditional DBMS with very large volume of RAM dedicated to cache data in order to increase performance
- Systems of copying data into the memory on the layer of logical scheme (work together with traditional DBMS), but not on the layer of data storage structure
- Distributed DB systems in RAM, in which each data element is stored on multiple instances deployed on multiple nodes of the system (since the fault of all nodes is almost impossible)
- Data storage systems used only to read data (DB for analytical data processing)

2.4 Обладнання. Обчислювальні ресурси

2.4 Hardware. Computation resources

- У зв'язку зі зниженням темпів зростання продуктивності окремих пристроїв суттєво зріс інтерес до паралельної обробки даних. Основними характеристиками якості паралельних систем є: прискорення, масштабованість за пропускнуою здатністю та масштабованість за часом відгуку.
- Due to decrease of performance growth of standalone devices, parallel data processing got significant interest recently. Main quality measures of parallel systems are: acceleration, scalability in throughput, and scalability in response time.

2.4 Обладнання. Обчислювальні ресурси

2.4 Hardware. Computation resources

Класи паралельних СУБД:

- SM (Shared Memory) – багатопроцесорні системи зі спільною оперативною пам'яттю і дисками, не потребують обміну даними для виконання операцій на різних процесорах
- SD (Shared Disks) – системи, де кожен процесор має свою оперативну пам'ять, недоступну іншим процесорам, але диски є спільними
- SN (Shared Nothing) – системи без розподілу якихось пристроїв, взаємодія паралельних частин відбувається через обчислювальну мережу

2.4 Обладнання. Обчислювальні ресурси

2.4 Hardware. Computation resources

Types of parallel DBMS:

- SM (Shared Memory) – multiprocessor systems with shared RAM and drives, do not require data exchange to execute operations on multiple processors
- SD (Shared Disks) – systems in which each processor has its own RAM, inaccessible by other processors, but with shared disks
- SN (Shared Nothing) – systems without sharing of any devices, interaction of parallel parts is performed through the network

2.4 Обладнання. Обчислювальні ресурси

2.4 Hardware. Computation resources

- Багатоядерні процесори мають спільний кеш, тому алгоритми паралельного виконання ефективно працюють при малих обсягах даних – для OLTP, але не для OLAP. Системи SN є паралельними та створюються для підвищення продуктивності, тоді як метою розподілених систем є підвищення доступності та відмовостійкості.
- Multicore processors have shared cache, then parallel algorithms are efficient for small data volumes – for OLTP but not for OLAP. SN systems are parallel and serve to increase performance, while distributed systems are used to increase accessibility and fault tolerance.

2.5 Сховища даних

2.5 Data warehouses

- Слабкі сторони традиційних СУБД (продуктивність та масштабованість) призвели до появи NoSQL систем. Такі БД дозволяють отримати високу продуктивність за рахунок ослаблення обмежень цілісності, більш слабкої підтримки транзакцій та відмови від високорівневих мов запитів (хоча багато NoSQL СУБД мають обмежені підмножини SQL).
- Weaknesses of traditional DBMS (performance and scalability) led to appearance of NoSQL systems. Such DB allow to get high performance because of relaxed integrity constraints, transactions support, and no high-level query language usage (however, many NoSQL DBMS support limited subsets of SQL).

2.5 Сховища даних

2.5 Data warehouses

Загальні характеристики NoSQL систем:

- Не використовують схему даних. Відмова від незалежності даних та програм за рахунок використання засобів синхронізації даних (обміну повідомленнями) на рівні програм.
- Слабко підтримують транзакційну семантику. Задачі підтримки узгодженості БД вирішуються на рівні програми, а не СУБД.
- Пристосовані до використання на кластерах. Простота структур зберігання даних та відсутність взаємозв'язків між елементами даних на рівні СУБД усуває необхідність координації при обробці даних, розміщених на різних серверах.

2.5 Сховища даних

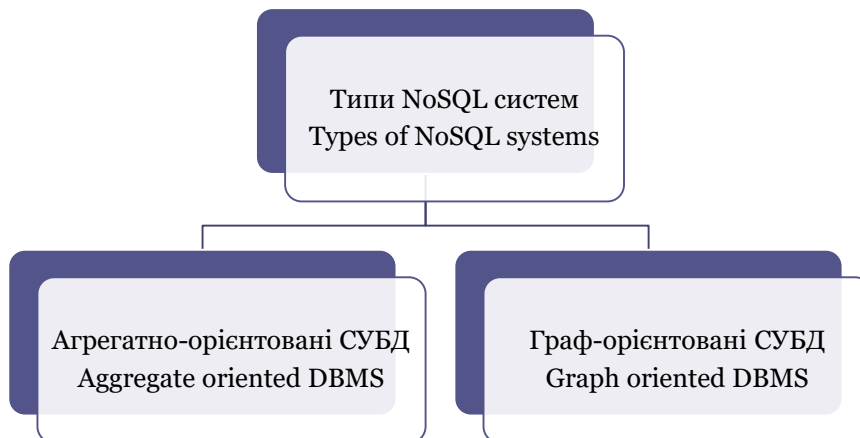
2.5 Data warehouses

General characteristics of NoSQL systems:

- Do not use data scheme. Refusal of independence of data and applications by using data synchronization tools (message exchange) on application layer.
- Weak support of transactions semantics. DB consistency is controlled on the application layer instead of DBMS layer.
- Suited for clustered usage. Simplicity of data storage structures and absence of relations between data elements on the DBMS layer eliminates necessity of coordination of data processing, which is deployed on multiple servers.

2.5 Сховища даних

2.5 Data warehouses



2.5 Сховища даних

2.5 Data warehouses

Агрегатно-орієнтовані NoSQL СУБД:

- Системи ключ-значення (Redis, Memcached). Значення може бути будь-яким (наприклад, рядок, документ або зображення). Сховище зберігає значення у двійковому вигляді, не знає про його структуру та не контролює тип.
- Документоорієнтовані системи (MongoDB, CouchDB). Зберігають довільні структури даних, як правило у форматі JSON. Підтримують запити на вибірку з пошуком за полями.
- Сховища сімейств колонок (HBase, Cassandra). Дані зберігаються у вигляді розрідженої матриці, ключами є строки та стовпці. Використовуються для OLAP систем.

Об'єкти можуть мати складну структуру, проте зберігаються як агрегати, не пов'язані між собою, тому їх нескладно розподілити за вузлами кластеру.

2.5 Сховища даних

2.5 Data warehouses

Aggregate oriented NoSQL DBMS:

- Key-value systems (Redis, Memcached). Value can be anything (e.g., string, document, or image). Warehouse stores values in binary format, does not know values structure and does not control value types.
- Document oriented systems (MongoDB, CouchDB). Store raw data structures, usually in JSON format. Support select queries with filtering by fields.
- Wide column store (HBase, Cassandra). Data is stored in the sparse matrix, which rows and columns are used as keys. Used for OLAP systems.

Objects could be of complex structure, while stores as aggregates that are not related with each other, therefore it is easy to distribute such objects to cluster nodes.

2.5 Сховища даних

2.5 Data warehouses

- Граф-орієнтовані СУБД. Зберігаються ребра, що демонструють зв'язок між об'єктами. Використовуються в рекомендаційних системах або соціальних мережах. Для вибірки підграфів використовуються спеціальні мови запитів. Підтримують транзакції на відміну від агрегатно-орієнтованих СУБД.
- Graph-oriented DBMS. Store edges that demonstrate relation between objects. Used in recommending systems or in social networks. Special query languages are used to select sub-graphs. Support transactions unlike aggregate oriented DBMS.

2.6 Вибір СУБД для побудови програм

2.6 Selecting DBMS for application development

- Функціональні вимоги. Усі сучасні СУБД підтримують мову SQL (Structured Query Language). Слід звернути увагу на необхідність нестандартних типів даних (XML, JSON, геолокації, часові інтервали), повнотекстового пошуку, багатомовність, підтримка збережених процедур та тригерів.
- Functional requirements. All modern DBMS support SQL (Structured Query Language) language. But you need to pay attention on need of non-standard data types (XML, JSON, geospatial, time series), full text search, multilanguage support, triggers and stored procedures support.

2.6 Вибір СУБД для побудови програм

2.6 Selecting DBMS for application development

- Нефункціональні вимоги. Вимоги до продуктивності, масштабованості, безпеки, резервного копіювання та відновлення після відмов.
- Non functional requirements. Requirements to performance, scalability, security, backup and restore after faults.

2.6 Вибір СУБД для побудови програм

2.6 Selecting DBMS for application development

- **Можливості для супроводу системи.** Супровід включає інсталяцію і налаштування системи, оновлення програмних компонент, забезпечення безперебійного функціонування. Необхідно звернути увагу на інструменти адміністрування СУБД та оцінити складність задач супроводу.
- **Maintenance capabilities.** Maintenance includes installation and configuration of the system, update of software components, ensuring of uninterrupted functioning. It is necessary to pay attention on administration tools of DBMS and estimate complexity of maintenance tasks.

2.6 Вибір СУБД для побудови програм

2.6 Selecting DBMS for application development

- **Бюджет проекту.** Супровід систем на основі високопродуктивних та високонадійних комерційних СУБД також дорогий та технічно складний. Необхідно враховувати наскільки складно знайти спеціалістів необхідної кваліфікації та рівень оплати їх праці.
- **Project budget.** Maintenance of systems based on high-performance and highly reliable DBMS is also expensive and technically complex. It is need to consider how hard is to hire specialists of required qualification and their payment rates.

2.6 Вибір СУБД для побудови програм

2.6 Selecting DBMS for application development

Тенденції використання СУБД:

- Відмова від використання функціональності СУБД. Перенос на рівень програми відповідальності за виконання складних операцій обробки даних, підтримки узгодженості та цілісності, обміну даними з іншими системами.
- Використання СУБД з обмеженими характеристиками або NoSQL систем. Обрана СУБД повністю задовольняє вимогам проекту або необхідні характеристики досягаються застосуванням апаратних ресурсів та обмеженням функціональності системи.
- Використання реляційних СУБД з відкритим кодом. Висока якість продукту та документації, розповсюдженість і активні спільноти роблять такі СУБД підходящими як для малих підприємств, так і для великих корпорацій.

2.6 Вибір СУБД для побудови програм

2.6 Selecting DBMS for application development

DBMS usage trends:

- Refusal of using of DBMS functions. Moving to application layer responsibility of execution of complex data processing operations, support of consistency and integrity, data exchange with other systems.
- Using DBMS with limited features or NoSQL systems. Selected DBMS completely satisfies project requirements, otherwise required features are achieved by using hardware resources and systems features constraints.
- Using open-source relational DBMS. High quality of product and documentation, popularity and active communities make such DBMS suitable for small organizations and big corporations.

Контрольні питання

1. Основні відмінності OLTP та OLAP систем.
2. Критерії оцінки якості паралельних систем.
3. Характеристики систем NoSQL.
4. Фактори вибору СУБД для розробки інформаційних систем.

Assessment questions

1. Main differences between OLTP and OLAP systems.
2. Evaluation criteria of parallel systems.
3. Characteristics of NoSQL systems.
4. DBMS selection factors for information systems development.

Моделі даних Data models

Тема 3
Торіс 3

3 Моделі даних 3 Data models

- Модель даних – система взаємопов'язаних понять і правил, призначена для опису структур і властивостей даних, що використовуються у БД.
- Data model is a system of interrelated definitions and rules, aimed to describe structures and data properties used in DB.
- Модель даних задає спосіб опису схеми БД.
- Data model defines the way to describe DB scheme.

3 Моделі даних

3 Data models

До складу моделі даних входять:

- Способи опису даних (базові типи та складні структури)
- Способи опису взаємозв'язків між об'єктами даних
- Засоби визначення обмежень цілісності
- Способи конструювання операцій

Data model includes:

- Ways to describe data (primitive types and complex structures)
- Ways to describe relations between data objects
- Tools to define integrity constraints
- Ways to build operations

3.1 Моделі даних. Ідентифікація та змінюваність

3.1 Data models. Identification and mutability

- Деякі моделі передбачають наявність виділеного способу ідентифікації (наприклад, об'єктний ідентифікатор у ОО моделях даних). Об'єкти вважаються співпадаючими, якщо в них однакові ідентифікатори. Елементи даних у складі об'єкту визначають його стан і такі об'єкти є змінюваними.
- Some models assume existence of identification mechanism (e.g. object identifier in OO models). Objects are equal if they have equal identifiers. Data elements of the object define its state, such objects are mutable.

3.1 Моделі даних. Ідентифікація та змінюваність

3.1 Data models. Identification and mutability



3.1 Моделі даних. Ідентифікація та змінюваність

3.1 Data models. Identification and mutability

- **За природними ознаками** об'єкта реального світу. Наприклад: код аеропорту, номер бронювання, біометричні властивості.
 - **За штучним значенням**, згенерованим ІС. Наприклад: номер документа ідентифікації особистості.
 - **За зв'язком об'єкта** з іншим об'єктом. Для розрізнення об'єктів, які неможливо розрізнити іншим способом. Наприклад: праве переднє колесо автомобілю ідентифікує його за місцем.
- **By natural features** of the real-world object. E.g. airport code, booking number, biometric properties.
 - **By artificial value** generated by IS. E.g. identity document number.
 - **By object relation** to another object. Used to identify objects that could not be identified in another way. E.g. right front wheel of the vehicle is identified by its place.

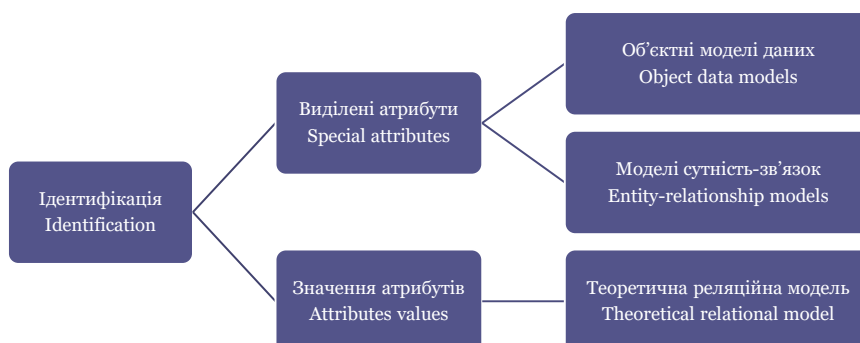
3.1 Моделі даних. Ідентифікація та змінюваність

3.1 Data models. Identification and mutability

- Якщо спосіб ідентифікації не визначено, необхідно перевіряти збіг значень усіх елементів даних. У таких моделях об'єкти з хоча б одним відмінним елементом є різними і тому не можуть змінюватися. Можна замінити об'єкт, але не змінювати.
- If the identification method is not specified, you must check that the values of all data elements match. In such models, objects with at least one different element are considered as different and therefore cannot be modified. You can replace the object, but not modify it.

3.1 Моделі даних. Ідентифікація та змінюваність

3.1 Data models. Identification and mutability



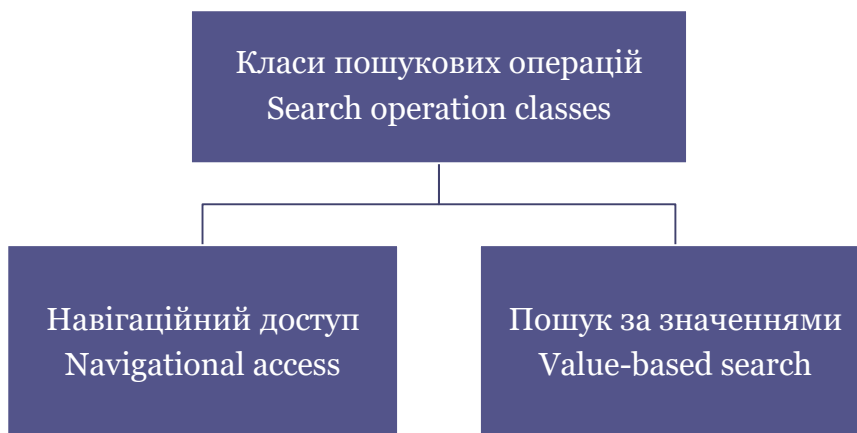
3.1 Моделі даних. Ідентифікація та змінюваність

3.1 Data models. Identification and mutability

- Внаслідок незмінюваності реляційна модель дозволяє побудувати потужні засоби виконання запитів. В деяких моделях ідентифікатори можуть бути виявлені на основі обмежень цілісності. Це використовується у реляційній моделі для визначення поняття ключа. В реальності практично усі об'єкти є змінюваними, тому важливим є вибір способу ідентифікації, що однозначно визначатиме об'єкти реального світу у БД.
- Due to the immutability the relational model allows you to build powerful tools for executing queries. In some models, identifiers can be detected based on integrity constraints. This is used in a relational model to define the concept of a key. In reality, almost all objects are mutable, so it is important to choose a method of identification that will uniquely identify real-world objects in the DB.

3.2 Моделі даних. Навігація та пошук за значеннями

3.2 Data models. Navigation and search by values



3.2 Моделі даних. Навігація та пошук за значеннями

3.2 Data models. Navigation and search by values

- Навігаційний пошук (переходи між об'єктами відбуваються за допомогою посилань). Прикладом є використання URL для навігації у Інтернеті. Використовуються явно задані зв'язки між об'єктами. Цей тип пошуку характерний для об'єктно-орієнтованих моделей даних та мов програмування.
- Navigational search (transitions between objects are made by links). An example is the use of URLs to navigate on the Internet. Explicitly defined relationships between objects are used. This type of search is typical for object-oriented data models and programming languages.

3.2 Моделі даних. Навігація та пошук за значеннями

3.2 Data models. Navigation and search by values

- Пошук по значенням (асоціативний пошук). Прикладом є пошукові системи, проте вони не обмежуються текстовим пошуком. Результатом є набір значень, що задовольняють умовам пошуку. Реалізується у декларативних мовах запитів, у теоретичній моделі даних, реальних СУБД на основі мови SQL.
- Search by values (associative search). Examples are search engines, but they are not limited to text search. The result is a set of values that satisfy the search conditions. Implemented in declarative query languages, in a theoretical data model, real DBMS based on SQL language.

3.2 Моделі даних. Навігація та пошук за значеннями

3.2 Data models. Navigation and search by values

- Навігація можлива навіть якщо модель даних не передбачає явну ідентифікацію об'єктів. Наприклад, можливо знайти колекцію взаємопов'язаних об'єктів. Не існує чіткого розрізнення між способами пошуку, багато моделей допускають обидва типи операцій пошуку.
- Navigation is possible even if the data model does not explicitly identify objects. For example, you might find a collection of related objects. There is no clear distinction between search methods, many models allow both types of search operations.

3.3 Моделі даних. Об'єкти та колекції об'єктів

3.3 Data models. Objects and object collections

- Визначені у моделі даних операції можуть бути орієнтовані на обробку окремих об'єктів та масову обробку. Зазвичай навігаційний доступ передбачає обробку окремих об'єктів даних, а пошук по значенням – масову обробку. Це повинно враховуватися при проектуванні програмного забезпечення.
- The operations defined in the data model can be focused on the processing of individual objects and mass processing. Typically, navigational access involves the processing of individual data objects, and the search for values involves bulk processing. This should be taken into account when designing the software.

3.3 Моделі даних. Об'єкти та колекції об'єктів

3.3 Data models. Objects and object collections

- Системи, що орієнтовані на обробку окремих об'єктів, є неефективними при масовій обробці. Системи, що демонструють високу продуктивність при масовій обробці, можуть бути вкрай неефективними при обробці окремих об'єктів. Теоретична реляційна модель та реляційні СУБД орієнтовані на масову обробку об'єктів.
- Systems that focus on processing individual objects are inefficient in mass processing. Systems that demonstrate high performance in bulk processing can be extremely inefficient in processing individual objects. Theoretical relational model and relational DBMS are focused on mass processing of data objects.

3.4 Моделі даних. Властивості моделей даних

3.4 Data models. Data models features

- Розглянуті властивості є достатньо абстрактними, але вони суттєво впливають на якість проєктованих систем. Ці властивості важливі для порівняльного аналізу моделей. Для ранніх моделей, включаючи об'єктні моделі, були характерні навігаційний спосіб доступу та орієнтація на обробку окремих об'єктів.
- The considered properties are rather abstract, but they essentially influence quality of the designed systems. These properties are important for comparative analysis of models. Early models, including object models, were characterized by a navigational access method and a focus on processing individual objects.

3.4 Моделі даних. Властивості моделей даних

3.4 Data models. Data models features

- Протягом років стало зрозумілим, що орієнтація на обробку окремих записів не може забезпечити ефективну масову обробку даних. З'явилися технології та реалізації реляційних систем, які значно випереджали ранні системи за продуктивністю. Тенденція відмови від масової обробки об'єктів на рівні БД призвела до розповсюдження NoSQL систем.
- Over the years, it has become clear that focusing on the processing of individual records cannot provide effective mass data processing. Technologies and implementations of relational systems have emerged that are far ahead of early systems in terms of performance. The trend of avoiding mass processing of objects at the database level has led to the spread of NoSQL systems.

Контрольні питання

1. Моделі даних та їх основні особливості.
2. Ідентифікація та змінюваність.
3. Навігація та пошук за значеннями.
4. Об'єкти та колекції об'єктів.
5. Властивості моделей даних.

Assessment questions

1. Data models and their main features.
2. Identification and mutability.
3. Navigation and search by values.
4. Objects and object collections.
5. Data models features.

Реляційна модель даних Relational data model

Тема 4
Topic 4

4 Реляційна модель даних

4 Relational data model

- Реляційна модель з'явилась на початку 1970-х рр. у працях Е. Кодда та інших дослідників. Десятки років розроблялася як суто теоретичний спосіб опису колекцій даних та мов запитів. Дослідження методів і структур зберігання та пошуку даних, оптимізації запитів, дозволили створити СУБД з високою ефективністю масової обробки даних.
- The relational model appeared in the early 1970s in the works of E. Codd and other researchers. It has been developed for decades as a purely theoretical way of describing data collections and query languages. Research of methods and structures of data storage and retrieval, query optimization, allowed to create DBMS with high efficiency of mass data processing.

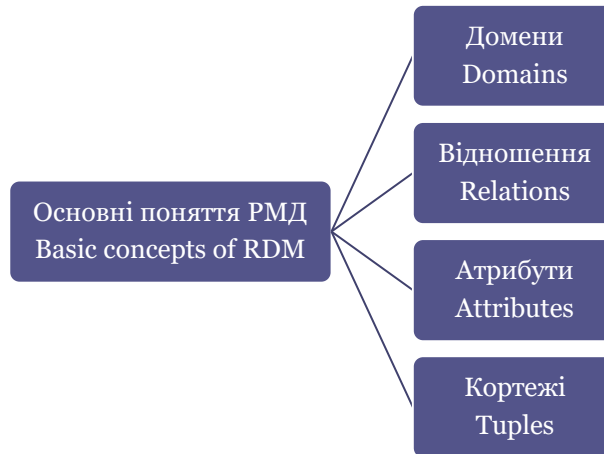
4 Реляційна модель даних

4 Relational data model

- Саме зміна критеріїв ефективності та вимог до систем (з 1980-х років найбільш важливою за доступ до окремих об'єктів та навігацію між ними стала масова обробка даних та пошук по значенням атрибутів) призвела до створення високоефективних систем на основі реляційної моделі даних.
- It is the change in performance criteria and system requirements (since the 1980s, the most important than accessing and navigating individual objects has become mass data processing and searching for attribute values) that has led to the creation of highly efficient systems based on a relational data model.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts



4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Домен можна описати як множину значень, що володіють деякими загальними властивостями. Домени цілих чисел, дійсних чисел, текстів тощо. Передбачається, що усі значення доменів є скалярними. Для кожної пари значень домену визначено відношення рівності. Це необхідно для порівняння, наприклад з константою у запиті.
- A domain can be described as a set of values that have some common properties. Domains of integers, real numbers, texts, etc. All domain values are assumed to be scalar. An equality relationship is defined for each pair of domain values. This is necessary for comparison, for example with a constant in the query.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Для доменів можуть бути визначені інші відношення, операції та функції. Наприклад у домені числових значень можна розглядати: відношення впорядковування ($<$, $<=$, $>$, $>=$), арифметичні операції ($+$, $-$, $*$, $/$), функції, що приймають скалярні значення.
- Other relationships, operations, and functions can be defined for domains. For example, in the domain of numerical values we can consider: ordering relations ($<$, $<=$, $>$, $>=$), arithmetic operations ($+$, $-$, $*$, $/$), functions that take scalar values.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Також можуть бути визначені функції, які приймають значення з інших доменів, функції декількох аргументів різних доменів. Наприклад, у домені текстів можна розглядати функцію, яка повертає довжину тексту. Додаткові відношення, функції та операції використовуються для обчислення нових значень при описі запитів, однак для реляційної моделі необхідним є лише відношення рівності.
- Functions that take values from other domains, functions of several arguments of different domains can also be defined. For example, in a text domain, you can consider a function that returns the length of the text. Additional relationships, functions, and operations are used to calculate new values when describing queries, but only an equality relationship is required for a relational model.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- У мовах програмування поняттю домену відповідає поняття абстрактного типу даних. Відмінність полягає у тому, що значення доменів є скалярними (тобто не можуть бути масивами, тощо). Домени є взаємопов'язаними (наприклад, домен цілих чисел є підмножиною домену дійсних чисел). Проте в реляційній теорії усі домени розглядаються як незалежні.
- In programming languages, the concept of domain corresponds to the concept of abstract data type. The difference is that domain values are scalar (i.e. cannot be arrays, etc.). Domains are interconnected (for example, the domain of integers is a subset of the domain of real numbers). However, in relational theory, all domains are considered independent.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Ранні реалізації реляційної моделі допускали використання лише обмеженого набору доменів: числові, дати та час, текстові рядки тощо. Сучасні СУБД дозволяють використання будь-яких доменів. Наприклад, можна визначити домени довжин та ваг. Вони будуть числовими, але набори операцій будуть відрізнятися. Наприклад, множення довжин даватиме результат у іншому домені, можливо, площин.
- Early implementations of the relational model allowed the use of only a limited set of domains: numeric, date and time, text strings etc. Modern databases allow the use of any domain. For example, you can define length and weight domains. They will be numeric, but the sets of operations will be different. For example, multiplying lengths will result in another domain, possibly squares.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Центральним поняттям реляційної моделі є відношення. Формально відношення визначається як n -арний предикат – функція з n аргументами, яка приймає булеве значення. Як і в математиці визначення предикату не передбачає наявності формули або алгоритму обчислення його значення.
- The central concept of the relational model is the relation. Formally, the relation is defined as an n -ary predicate – a function with n arguments, which takes a boolean value. As in mathematics, the definition of a predicate does not require a formula or algorithm for calculating its value.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- У реляційній теорії використовуються іменовані аргументи, які називаються атрибутами відношення. Використання імен дозволяє записувати атрибути у довільному порядку. З кожним атрибутом пов'язаний домен його значень. Один домен може бути пов'язаний з декількома атрибутами. Множина усіх атрибутів відношення називається схемою відношення.
- Relational theory uses named arguments called relationship attributes. Using names allows you to write attributes in any order. Each attribute is associated with a domain of its values. A single domain can be associated with multiple attributes. The set of all attributes of a relation is called a relation scheme.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Сукупність n значень по одному для кожного атрибуту називається кортежем. Кортеж – набір значень аргументів, для якого можна обчислити предикат, що відповідає відношенню. Вважається, що значення у кортежі взаємозалежні, якщо цей кортеж належить відношенню та виражає істинність деякого факту про відповідний об'єкт реального світу.
- The set of n values, one for each attribute, is called a tuple. A tuple is a set of argument values for which a predicate corresponding to a relation can be computed. Values in a tuple are considered to be interdependent if the tuple belongs to a relation and expresses the truth of some fact about the corresponding real-world object.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Кількість кортежів у деякому відношенні називається кардинальністю цього відношення. Наприклад, відношення exams має атрибути {name, course, grade}. Тоді істинне значення предиката на наступних кортежах вказує на те, що зазначені студенти отримали оцінки по даних курсах:
- The number of tuples in some relation is called the cardinality of this relation. For example, the exams relationship has the attributes {name, course, grade}. Then the true value of the predicate for the following tuples indicates that these students received grades for these courses:

```
exams(name='Ann', course='Databases', grade=5)
exams(name='Ann', course='Data analysis', grade=5)
exams(course='Data analysis', grade=5, name='Victor')
exams(name='Nina', grade=5, course='Databases')
```

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Кожний кортеж складається з n значень, тому відношення прийнято записувати у вигляді таблиць з n колонок, які відповідають атрибутам відношення, а кожний рядок відповідає одному кортежу. Порядок розташування кортежів не має значення.
- Each tuple consists of n values, so the relationship is usually written in the form of a table with n columns that correspond to the attributes of the relationship, and each row corresponds to one tuple. The order of the tuples does not matter.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- Кожний кортеж визначає істинність деякого факту, тому повторне включення кортежу до відношення не дає ніякої нової інформації. У теоретичній реляційній моделі передбачається, що усі кортежі є різними і тому сукупність кортежів відношення є множиною у математичному сенсі.
- Each tuple determines the truth of a fact, so the re-inclusion of the tuple in the relationship does not provide any new information. The theoretical relational model assumes that all tuples are different and therefore the set of tuples of the relation is a set in the mathematical sense.

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

exams

name	course	grade
Ann	Databases	5
Ann	Data analysis	5
Victor	Data analysis	4
Nina	Databases	5

4.1 Реляційна модель даних. Основні поняття

4.1 Relational data model. Basic concepts

- У багатьох реалізаціях реляційної моделі даних відношення називаються таблицями, атрибути – колонками або стовпцями, а кортежі – рядками. На відміну від теоретичної реляційної моделі, табличні реалізації у СУБД допускають зберігання співпадаючих рядків у таблицях, якщо не задано унікальність.
- In many implementations of the relational data model, relationships are called tables, attributes are called columns, and tuples are called rows. In contrast to the theoretical relational model, tabular implementations in the DBMS allow the storage of equal rows in tables, unless unique constraint is defined.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- Успіх реляційної моделі даних пояснюється потужними та виразними мовами маніпулювання даними, які вона визначає. Результати виконання операцій можна використовувати як аргументи наступних операцій, задаючи обчислення у формі виразів. Такі системи операцій називаються алгебрами. Набір операцій на множині всіх можливих кінцевих відносин називається реляційною алгеброю.
- The success of the relational data model is due to the powerful and expressive data manipulation languages it defines. The results of operations can be used as arguments for subsequent operations, specifying calculations in the form of expressions. Such systems of operations are called algebras. A set of operations on the set of all possible finite relations is called relational algebra.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- **Теоретико-множинні операції.** Можуть бути застосовані до будь-якої пари відношень, які мають однакові схеми.
- **Sets theory operations.** Can be applied to any pair of relationships that have the same scheme.

UNION – об'єднання, union operation, объединение

$$\{a, b\} \text{ UNION } \{a, c\} = \{a, b, c\}$$

INTERSECT – перетин, intersect operation, пересечение

$$\{a, b\} \text{ INTERSECT } \{a, c\} = \{a\}$$

EXCEPT (MINUS) – різниця, except operation, вычитание

$$\{a, b\} \text{ EXCEPT } \{a, c\} = \{b\}$$

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- Ці операції можна використовувати для побудови похідних операцій, визначених алгебраїчними виразами. Наприклад, операцію симетричної різниці відношень R та S (XOR) можна визначити як:
- These operations can be used to construct derivative operations defined by algebraic expressions. For example, the operation of the symmetric difference of the relations R and S (XOR) can be defined as:

$$(R \cup S) \text{ EXCEPT } (R \cap S)$$

або як:

or as:

$$(R \text{ EXCEPT } S) \cup (S \text{ EXCEPT } R)$$

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- **Унарні операції.** Операції з одним аргументом дозволяють виділити необхідну інформацію.
- **Unary operations.** Operations with one argument allow you to select the necessary information.
- Операція проєкції PROJ включає у результат підмножину атрибутів відношення. Наприклад PROJ [name, course] exams дозволяє отримати лише дані про факт екзамену без оцінки.
- The PROJ projection operation results in a subset of the relationship attributes. For example PROJ [name, course] exams allows you to get only the data on the fact of the exam without a grade.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

PROJ [name, course] exams

name	course
Ann	Databases
Ann	Data analysis
Victor	Data analysis
Nina	Databases

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- В результаті виключення частини атрибутів різні кортежі вихідного відношення стануть співпадати за значенням. Реляційна операція проекції обов'язково виключить дублікати. Проекція на атрибути {course, grade} буде містити менше кортежів, ніж вихідне відношення.
- As a result of exclusion of a part of attributes various tuples of initial relation will begin to coincide on value. The relational projection operation will eliminate duplicates anyway. The projection on the {course, grade} attributes will contain fewer tuples than the original relationship.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

PROJ [course, grade] exams

course	grade
Databases	5
Data analysis	5
Data analysis	4

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- Операція фільтрації FILTER будує нове відношення, включаючи в нього рядки вихідного відношення, які задовольняють умові, що виражається логічною формулою. Проста умова представляється атрибутом, який порівнюється з константою або іншим атрибутом. Умови можна поєднувати логічними операторами (AND, OR, NOT).
- The FILTER filtering operation builds a new relationship, including the lines of the original relationship that satisfy the condition expressed by the logical formula. A simple condition is represented by an attribute that is compared to a constant or other attribute. Conditions can be combined with logical operators (AND, OR, NOT).

123

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

exams

name	course	grade
Ann	Databases	5
Ann	Data analysis	5
Victor	Data analysis	4
Nina	Databases	5

FILTER [course='Data analysis' AND grade=5] exams

name	course	grade
Ann	Data analysis	5

124

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- **Добуток.** Операція декартового добутку PROD будує усі пари кортежів з першого та другого аргументів та створює кортеж результату для кожної такої пари. Усі імена атрибутів повинні бути різними, тому перед виконанням операції атрибути з однаковими іменами у одному з відношень потрібно перейменувати.
- **Product.** The Cartesian product PROD constructs all pairs of tuples from the first and second arguments and creates a result tuple for each such pair. All attribute names must be different, so you must rename attributes with the same name in one of the relationships before performing the operation.

125

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

exams

name	course	grade
Ann	Databases	5
Ann	Data analysis	5
Victor	Data analysis	4
Nina	Databases	5

courses

title	credits
Databases	5
Data analysis	10

126

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

exams PROD courses

name	course	grade	title	credits
Ann	Databases	5	Databases	5
Ann	Data analysis	5	Databases	5
Victor	Data analysis	4	Databases	5
Nina	Databases	5	Databases	5
Ann	Databases	5	Data analysis	10
Ann	Data analysis	5	Data analysis	10
Victor	Data analysis	4	Data analysis	10
Nina	Databases	5	Data analysis	10

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- Кардинальність добутку відношень дорівнює добутку кардинальностей аргументів. Безпосередньо добуток не є корисною операцією, оскільки результат містить кортежі, які представляють собою істинні але не обов'язково пов'язані між собою факти.
- The cardinality of the product of relations is equal to the product of the cardinalities of the arguments. The product itself is not a useful operation because the result contains tuples that are true but not necessarily related facts.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- **З'єднання.** Операція JOIN представляє собою з'єднання з подальшою фільтрацією. Умова фільтрації може містити та зіставляти атрибути з різних аргументів, виділяючи таким чином взаємопов'язані кортежі з різних відношень.
- **Join.** The JOIN operation is a join followed by filtering. The filter condition can contain and compare attributes from different arguments, thus distinguishing interrelated tuples from different relationships.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- Умова фільтрації може бути будь-якою, проте зазвичай використовуються умови, які представляють собою кон'юнкцію умов на рівність значень пар атрибутів. Значення порівнюваних атрибутів у результуючому відношенні будуть однаковими, тому необхідно зробити проекцію, яка виключить дублюючі значення.
- The filtering condition can be any, but usually used are conjunctions of conditions on the equality of values of attribute pairs. The values of the compared attributes will be the same in the result set, so it is necessary to make a projection that will eliminate duplicate values.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

exams JOIN [course=title] courses

name	course	grade	title	credits
Ann	Databases	5	Databases	5
Nina	Databases	5	Databases	5
Ann	Data analysis	5	Data analysis	10
Victor	Data analysis	4	Data analysis	10

PROJ [name, course, grade, credits] (exams JOIN [course=title] courses)

name	course	grade	credits
Ann	Databases	5	5
Nina	Databases	5	5
Ann	Data analysis	5	10
Victor	Data analysis	4	10

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- Операція з'єднання є похідною, оскільки вона виражається через добуток та фільтрацію. Вона розглядається окремо, оскільки вона дуже важлива для застосування. Важливо також, що для операції з'єднання існують більш ефективні алгоритми, ніж алгоритми, засновані на прямому добутку. Інші варіанти будуть розглянуті в контексті мови запитів SQL.
- The join operation is derived because it is expressed through product and filtering. It is considered separately because it is very important for application. It is also important that there are more efficient algorithms for the join operation than direct product-based algorithms. Other options will be considered in the context of the SQL query language.

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

Властивості операцій реляційної алгебри:

- Комутативність. Операції об'єднання, перетину, добутку та з'єднання.

$$R \cup S = S \cup R$$

$$R \Join S = S \Join R$$

- Асоціативність. Операції перетину, об'єднання, добутку та з'єднання.

$$R \cup (S \cup T) = (R \cup S) \cup T$$

$$R \Join (S \Join T) = (R \Join S) \Join T$$

- Дистрибутивність. Пари операцій об'єднання, перетину, добутку (або з'єднання).

$$R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$$

$$R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$$

$$R \cap (S \Join T) = (R \cap S) \Join (R \cap T)$$

$$R \Join (S \cap T) = (R \Join S) \cap (R \Join T)$$

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

Features of relational algebra operations:

- Commutative property. Union, intersect, product, and join operations.

$$R \text{ UNION } S = S \text{ UNION } R$$

$$R \text{ JOIN } S = S \text{ JOIN } R$$

- Associative property. Intersect, union, product, and join operations.

$$R \text{ UNION } (S \text{ UNION } T) = (R \text{ UNION } S) \text{ UNION } T$$

$$R \text{ JOIN } (S \text{ JOIN } T) = (R \text{ JOIN } S) \text{ JOIN } T$$

- Distributive property. Pairs of merge, intersection, product (or join) operations.

$$R \text{ INTERSECT } (S \text{ UNION } T) = (R \text{ INTERSECT } S) \text{ UNION } (R \text{ INTERSECT } T)$$

$$R \text{ UNION } (S \text{ INTERSECT } T) = (R \text{ UNION } S) \text{ INTERSECT } (R \text{ UNION } T)$$

$$R \text{ PROD } (S \text{ UNION } T) = (R \text{ PROD } S) \text{ UNION } (R \text{ PROD } T)$$

$$R \text{ PROD } (S \text{ INTERSECT } T) = (R \text{ PROD } S) \text{ INTERSECT } (R \text{ PROD } T)$$

4.2 Реляційна модель даних. Реляційна алгебра

4.2 Relational data model. Relational algebra

- Існування різних тотожностей дозволяє перетворювати алгебраїчні вирази у еквівалентні. Результат обчислення еквівалентних виразів буде однаковим, але складність може відрізнятися на декілька порядків. СУБД може обирати серед еквівалентних способів запису такий, що потребує меншої кількості ресурсів.
- The existence of different forms makes it possible to convert algebraic expressions into equivalent ones. The result of calculating equivalent expressions will be the same, but the complexity may differ by several orders. The DBMS can choose from equivalent expressions that require fewer resources.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

- Наявність мови запитів є однією з основних вимог до СУБД. Така мова має бути декларативною – описувати який треба отримати результат, не вказуючи спосіб його обчислення. Реляційна алгебра не є декларативною мовою, так як алгебраїчний вираз визначає порядок виконання операцій.
- The presence of the query language is one of the main requirements for the DBMS. Such language should be declarative – to describe what result it is necessary to receive, without specifying a way of its calculation. Relational algebra is not a declarative language, as an algebraic expression determines the order of operations.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

- Мови запитів більш високого рівня дозволяють записати вимоги до результату виконання запиту у вигляді набору логічних умов. Такі мови називаються обчисленнями. У лівій частині правила визначається схема відношення, яка є результатом обчислення правила, а у лівій – умова, якій повинні задовольняти кортежі, включені у результат обчислення.
- Higher-level query languages allow you to write requirements to the result of the query in the form of a set of logical conditions. Such languages are called calculus. The left part of the rule defines the scheme of the relationship, which is the result of the calculation of the rule, and the left – the condition that must satisfy the tuples included in the result of the calculation.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

Ліва частина правила називається *головою* правила, а права частина, яка містить умову, називається *тілом* правила.

Умова формується наступним чином:

- Прості умови задаються предикатами, визначеними на доменах; в якості аргументів предикатів можуть використовуватися змінні або константи.
- Умова може бути подана у дужках.
- Умова може бути побудована з більш простих умов за допомогою логічних операцій \wedge , \vee , \neg .
- Умова, яка містить вільні змінні, може бути замкнута за допомогою кванторів загальності \forall та існування \exists .
- Вільні змінні можуть використовуватися у лівій частині правила в якості атрибутів результуючого відношення.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

The left part of the rule is called the *head* of the rule, and the right part, which contains the condition, is called the *body* of the rule.

The condition is formed as follows:

- Simple conditions are given by predicates defined on domains; variables or constants can be used as arguments.
- The condition can be given in parentheses.
- The condition can be constructed from simpler conditions by using logical operations \wedge , \vee , \neg .
- A condition that contains free variables can be closed by quantifiers of generality \forall and existence \exists .
- Free variables can be used on the left side of the rule as attributes of the resulting relationship.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

- Розрізняють обчислення зі змінними на кортежах і на доменах. Для змінних на кортежах вказується приналежність змінної до відношення. Значення атрибутів позначаються ім'ям змінної, за яким слідує ім'я атрибуту, відокремлене крапкою. Голова і тіло розділяються знаком :-.
- There are calculations with variables on tuples and domains. For variables on tuples the affiliation of a variable to the relation is specified. Attribute values are denoted by the variable name, followed by the attribute name separated by a period. The head and body are separated by a sign :-.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

exams JOIN [course=title] courses

name	course	grade	title	credits
Ann	Databases	5	Databases	5
Nina	Databases	5	Databases	5
Ann	Data analysis	5	Data analysis	10
Victor	Data analysis	4	Data analysis	10

JoinResult (x.name, x.course, x.grade, y.title, y.credits) :-
 $x \in \text{exams} \wedge y \in \text{courses} \wedge x.\text{course} = y.\text{title}$

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

- У обчисленні на доменах змінні приймають значення у доменах. З таких змінних і констант формуються кортежі, приналежність яких до відношень БД є однією з простих умов у виразі, який описує запит. У прикладі тіло правила описує запит на вибірку результатів екзаменів з оцінкою, нижчою за відмінну.
- In calculus on domains, variables take values in domains. From such variables and constants tuples are formed, belonging to the relations of the database is one of the simple conditions in the expression that describes the query. In the example, the rule's body describes a request for exam results with a grade below excellent.

$$\{\text{name, course, grade}\} \in \text{exams} \wedge \text{grade} < 5$$

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

- У реляційному обчисленні не допускається рекурсія, тобто відношення, яке знаходиться в голові правила, не може використовуватися у його тілі, а якщо правил декілька, то не допускається також взаємна рекурсія.
- In relational computation, recursion is not allowed, i.e. the relation that is in the head of the rule cannot be used in its body, and if there are several rules, then mutual recursion is also not allowed.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

- Існує декілька варіантів реляційних обчислень, однак можна доказати, що усі вони еквівалентні одне одному і кожний з яких еквівалентний реляційній алгебрі. Еквівалентність означає, що будь-який запит, який можна записати на одній з цих мов, можна також записати і на іншій еквівалентній мові.
- There are several variants of relational calculus, but we can prove that they are all equivalent to each other and each of them is equivalent to relational algebra. Equivalence means that any query that can be written in one of these languages can also be written in another equivalent language.

4.3 Реляційна модель даних. Інші мови запитів

4.3 Relational data model. Other query languages

- Мова SQL, яка застосовується у реалізаціях СУБД, займає проміжне положення між алгеброю та обчисленням та дозволяє використовувати форми запису запитів, близькі як до алгебраїчних виразів, так і до формул в обчисленні. Незалежно від обраного способу запису при підготовці запиту до виконання він переводиться у алгебраїчний вираз. Серед еквівалентних обирається вираз, виконання якого буде обчислено ефективно.
- SQL, which is used in DBMS implementations, has an intermediate position between algebra and calculus and allows the use of query forms that are close to both algebraic expressions and formulas in computation. Regardless of the chosen method of writing when preparing a request for execution, it is translated into an algebraic expression. An expression is selected from the equivalent, the execution of which will be calculated efficiently.

4.4 Реляційна модель даних. Особливості реляційної моделі даних

4.4 Relational data model. Relational model features

- Семантика реляційної моделі визначається тим, що кортежі відповідають твердженням (фактам) про об'єкти реального світу. Кортежі виконують роль об'єктів даних. Формально кортежі не можуть бути змінені. Можна замінити одне значення на інше, але це буде інший кортеж.
- The semantics of the relational model are determined by the fact that tuples correspond to statements (facts) about real-world objects. Tuples act as data objects. Formally, tuples cannot be changed. You can replace one value with another, but it will be another tuple.

4.4 Реляційна модель даних. Особливості реляційної моделі даних

4.4 Relational data model. Relational model features

- Усі операції реляційної алгебри в якості результату створюють нове відношення, тому реляційна модель орієнтована на масову обробку даних. У реляційній алгебрі передбачені лише критерії пошуку. Взаємозв'язки між кортежами різних відношень також встановлюються у операціях з'єднання на основі значень атрибутів.
- All operations of relational algebra as a result create a new relationship, so the relational model is focused on mass data processing. Relational algebra provides only search criteria. Relationships between tuples of different relationships are also established using join operations based on attribute values.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Реляційна теорія аналізує різні типи залежностей.

Нехай X та Y – деякі множини атрибутів одного відношення. Y функціонально залежить від X ($X \rightarrow Y$), якщо для будь-якої комбінації значень атрибутів з X існує лише одна комбінація значень Y , яка входить у відношення.

- Relational theory analyzes different types of dependencies.

Let X and Y be some sets of attributes of one relation. Y is functionally dependent on X ($X \rightarrow Y$) if for any combination of attribute values of X there is only one combination of Y values that is included in the relationship.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Функціональна залежність означає, що існує деяка функція з областю визначення X , яка приймає значення у Y та визначає значення Y для будь-якого кортежу, що може входити у розглянуте відношення. Важливе лише існування такої функції, а не спосіб її обчислення.

- Functional dependence means that there is some function with a domain X that takes values in Y and determines the value of Y for any tuple that may be included in the relationship under consideration. What matters is only the existence of such a function, not the way it is calculated.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Якщо $Y \subset X$, то $X \rightarrow Y$, оскільки будь-яка комбінація значень атрибутів однозначно визначає саму себе. Такі залежності називаються *тривіальними* і існують для будь-якої множини атрибутів. Усі атрибути відношення залежать від схеми цього відношення та кожний атрибут залежить сам від себе.
- If $Y \subset X$, then $X \rightarrow Y$, because any combination of attribute values uniquely defines itself. Such dependencies are called *trivial* and exist for any set of attributes. All attributes of a relation depend on the scheme of this relation and each attribute depends on itself.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Нетривіальні залежності не можуть бути виведені формально, вони повинні відображати закономірності, яким підкоряються властивості об'єктів реального світу, представлених у СУБД. Такі закономірності повинні бути справедливими для будь-якого стану бази даних, тобто повинні виконуватися для усіх без виключення об'єктів реального світу.
- Non-trivial dependencies cannot be derived formally, they must reflect the laws that govern the properties of real-world objects represented in the database. Such patterns must be true for any state of the database, that is, they must be true for all real-world objects.

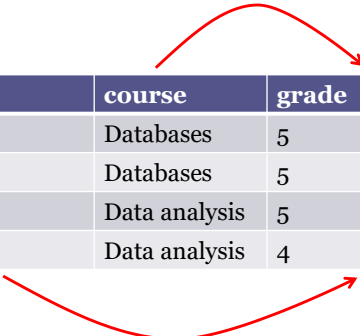
4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Наприклад, у відношенні *exams* оцінка (*grade*) функціонально залежить від пари атрибутів {*name*, *course*}, оскільки кожний студент може мати лише одну підсумкову оцінку за будь-якою дисципліною. Залежності між атрибутами *name* та *course* немає, так як будь-який курс може здавати декілька студентів і один студент може здавати декілька курсів. Ні *name*, ні *course* не залежать від оцінки навіть у комбінації з іншим атрибутом, тому {*name*, *course*} → {*grade*} є єдиною нетривіальною функціональною залежністю у цьому відношенні.
- For example, in relation *exams*, the *grade* functionally depends on the pair of attributes {*name*, *course*}, as each student can have only one final grade in each discipline. There is no relationship between the name and course attributes, as any course can be taken by several students and one student can take several courses. Neither *name* nor *course* depends on the *grade*, even in combination with another attribute, so {*name*, *course*} → {*grade*} is the only non-trivial functional dependency in this relation.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms



name	course	grade	credits
Ann	Databases	5	5
Nina	Databases	5	5
Ann	Data analysis	5	10
Victor	Data analysis	4	10

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Множина атрибутів, від якої функціонально залежать усі атрибути відношення, називається *потенційним ключем* відношення. У будь-якому відношенні є принаймні один потенційний ключ, оскільки будь-який атрибут тривіально залежить від усіх атрибутів відношення.
- The set of attributes on which all the attributes of a relation depend functionally is called the *potential key* of the relation. There is at least one potential key in any relationship, because any attribute is trivially dependent on all the attributes of the relationship.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Потенційний ключ називається *мінімальним ключем*, якщо після виключення з нього будь-якого атрибуту множина атрибутів, що залишилась, не є потенційним ключем. У одному відношенні може бути декілька мінімальних ключів, один з яких обирається в ролі *первинного ключа* відношення.
- A potential key is called a *minimum key* if, after excluding any attribute from it, the remaining set of attributes is not a potential key. In one relationship there may be several minimum keys, one of which is selected as the *primary key* of the relationship.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Значення первинного ключа, як і будь-якого потенційного, є унікальними та можуть використовуватися для ідентифікації об'єктів реального світу, що описуються кортежами відношення. Різні кортежі з однаковими значеннями первинного ключа описують різні стани одного реального об'єкту. Звісно вони зберігаються в різних станах відношення у різні моменти часу реального світу.
- Primary key values, like any potential key, are unique and can be used to identify real-world objects described by relationship tuples. Different tuples with the same primary key values describe different states of the same real object. Of course, they persist in different states of relationship at different time points in the real world.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Наприклад, у відношенні *exams* єдиним мінімальним потенційним ключем є $\{name, course\}$. Ця пара атрибутів складає первинний ключ даного відношення.
- For example, for relationship *exams*, the only minimum potential key is $\{name, course\}$. This pair of attributes is the primary key of this relationship.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

name	course	grade	credits
Ann	Databases	5	5
Nina	Databases	5	5
Ann	Data analysis	5	10
Victor	Data analysis	4	10

PK

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms


- Наявність нетривіальних функціональних залежностей може призводити до небажаних ефектів, які називають *аномаліями*. Крім вже згаданої залежності $\{name, course\} \rightarrow \{grade\}$, у відношенні наявна залежність $\{course\} \rightarrow \{credits\}$. Через це присутня надлишковість: credits вказуються стільки разів, скільки разів зустрічається курс.
- The presence of non-trivial functional dependencies can lead to side effects, which are called *anomalies*. In addition to the already mentioned dependence $\{name, course\} \rightarrow \{grade\}$, there is a dependence $\{course\} \rightarrow \{credits\}$. Because of this there is a redundancy: credits are indicated as many times as the course occurs.

159

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

4.5 Реляционная модель данных. Нормальные формы



name	course	grade	credits
Ann	Databases	5	5
Nina	Databases	5	5
Ann	Data analysis	5	10
Victor	Data analysis	4	10

160

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Більш суттєвим недоліком є неможливість зберігання інформації про курс, який не здавав жоден студент. Включити інформацію про курс можна лише разом з оцінкою та ім'ям студента, а видалення усіх оцінок з курсу призведе до втрати інформації про курс.
- A more significant drawback is the inability to store information about a course that no student has taken. Course information can only be included with the student's grade and name, and removing all grades from the course will result in the loss of course information.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Причиною є те, що атрибут *credits* залежить лише від частини первинного ключа. Для усунення аномалій у даному випадку необхідно використовувати замість такого відношення дві його проекції, які співпадають з відношеннями *exams* та *courses*.
- The reason is that the *credits* attribute depends only on part of the primary key. To eliminate anomalies in this case, it is necessary to use instead of such a relationship two of its projections, which coincide with the relationship *exams* and *courses*.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

exams

name	course	grade
Ann	Databases	5
Ann	Data analysis	5
Victor	Data analysis	4
Nina	Databases	5

courses

title	credits
Databases	5
Data analysis	10

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Відношення, у яких усі атрибути мають скалярні значення, називаються відношеннями у *першій нормальній формі* (1NF). Відповідно до реляційної теорії, усі відношення знаходяться у 1NF.
Відношення, у яких відсутні залежності від неповного ключа, називаються відношеннями у *другій нормальній формі* (2NF).
- Relationships in which all attributes have scalar values are called relationships in the *first normal form* (1NF). According to relational theory, all relations are in 1NF.
Relationships in which there are no dependencies on the incomplete key are called relations in the *second normal form* (2NF).

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Аномалії також можуть бути викликані транзитивними залежностями. Якщо наявні функціональні залежності $X \rightarrow Y$ та $Y \rightarrow Z$, то існує ще і залежність $X \rightarrow Z$, яка є суперпозицією перших двох. Такі комбінації призводять до аномалій, оскільки кожний комплект значень атрибутів з Z буде повторений разом з відповідними значеннями атрибутів з Y .
- Anomalies can also be caused by transitive dependencies. If there are functional dependences $X \rightarrow Y$ and $Y \rightarrow Z$, then there is also the dependence $X \rightarrow Z$, which is a superposition of the first two. Such combinations lead to anomalies because each set of attribute values with Z will be repeated along with the corresponding attribute values with Y .

165

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Наприклад, у відношенні зі схемою {emp, dept, mgr}, для кожного співробітника вказаний його відділ і менеджер. У такому відношенні наявна надлишковість, оскільки у всіх співробітників одного відділу менеджер один і той же самий.
- For example, in relation with the scheme {emp, dept, mgr}, for each employee the department and manager are specified. There is a redundancy in this relation, because all employees in the same department have the same manager.

166

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

Redundancy



emp	dept	mgr
Jim Halpert	Sales	Michael Scott
Dwight Schrutte	Sales	Michael Scott
Pamela Bisley	Sales	Michael Scott
Kelly Kapour	HR	Toby Flenderson

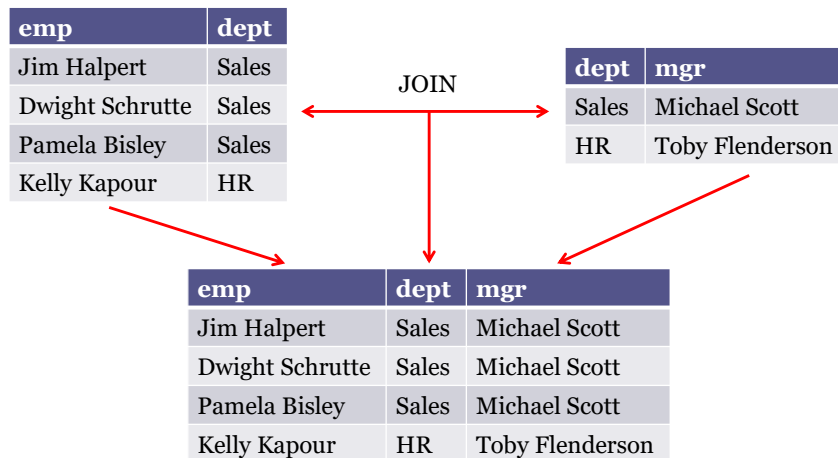
4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Для усунення таких аномалій вихідне відношення замінюється на його проекції таким чином, щоб транзитивні залежності опинилися у різних відношеннях. У прикладі такими відношеннями можуть бути {emp, dept} та {dept, mgr}. З'єднання цих проекцій відновить вихідне відношення, тому при створенні проекцій втрати інформації не відбувається.
- To eliminate such anomalies, the original relation is replaced by its projection so that the transitive dependences are in different relations. In the example, such relationships may be {emp, dept} and {dept, mgr}. Connecting these projections will restore the original relationship, so there is no loss of information when creating projections.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms



4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Усунення транзитивних залежностей переводить відношення у *третю нормальну форму* (3NF). Кажуть, що відношення знаходиться у *нормальній формі Бойса-Кодда* (BKNF), якщо для будь-якої нетривіальної функціональної залежності $X \rightarrow Y$ між атрибутами цього відношення множина X містить деякий ключ цього відношення. Різниця між 3NF та BKNF не є суттєвою з практичної точки зору.
- Elimination of transitive dependences translates the relationship into the *third normal form* (3NF). A relation is considered to be in the *Boyce-Codd normal form* (BKNF) if for any nontrivial functional dependence $X \rightarrow Y$ between the attributes of this relation the set X contains some key of this relation. The difference between 3NF and BKNF is not significant from a practical point of view.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Практичне значення нормальних форм та нормалізації полягає у тому, що вони дають критерії оцінки якості логічної структури бази даних, на основі якої будуються представлення структури даних для програм. Структури зберігання можуть відрізнятися від логічної структури, оскільки при їх проектуванні враховуються й інші критерії.
- The practical significance of normal forms and normalization is that they provide criteria for assessing the quality of the logical structure of the database, on the basis of which the representation of the data structure for programs is built. Storage structures may differ from the logical structure, because their design takes into account other criteria.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- Безпосереднє зберігання логічної схеми може призвести до необхідності частого виконання обчислювально складних операцій з'єднання у багатьох запитах. Щоб виключити зайві обчислення, у таких випадках варто організувати зберігання ненормалізованих відношень (на рівні схеми зберігання, а не на логічному рівні).
- Direct storage of a logic scheme may require frequent computational expensive join operations in many queries. To avoid unnecessary computation, in such cases it is necessary to organize the storage of non-normalized relationships (at the level of the storage scheme, not at the logical level).

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- У деяких випадках нормалізацією називають інші проектні рішення. Типовою є заміна атрибутів (наприклад рядкових) на сурогатні ідентифікатори з подальшим винесенням їх значень у окремі відношення. Таке рішення належить до нормалізації лише тим, що воно вводить штучні транзитивні залежності. В рамках реляційної моделі немає необхідності у додаткових сурогатних ідентифікаторах, які також призводять до ускладнення багатьох запитів.
- In some cases, normalization is associated with other design solutions. It is typical to replace attributes (for example, strings) with surrogate identifiers with subsequent removal of their values to separate relations. Such a solution belongs to normalization only in that it introduces artificial transitive dependencies. Within the relational model, there is no need for additional surrogate identifiers, which also complicate many queries.

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

dept_id	dept	mgr_id
1	Sales	1
2	HR	2

emp	dept_id
Jim Halpert	1
Dwight Schrutte	1
Pamela Bisley	1
Kelly Kapour	2

mgr_id	mgr
1	Michael Scott
2	Toby Flenderson

4.5 Реляційна модель даних. Нормальні форми

4.5 Relational data model. Normal forms

- У теоретичній реляційній моделі крім функціональних залежностей розглядається ціла низка інших класів залежностей і пов'язаних з ними нормальних форм. Наприклад, *багатозначні залежності* визначають відповідність між групами значень атрибутів. Усунення небажаних багатозначних залежностей призводить до *четвертої нормальної форми*. Однак ці типи залежностей та нормальні форми мають невелике практичне значення.
- In the theoretical relational model, in addition to functional dependencies, a number of other classes of dependencies and related normal forms are considered. For example, *multivalued dependencies* determine the correspondence between groups of attribute values. Elimination of unwanted multivalued dependencies leads to the *fourth normal form*. However, these types of dependencies and normal forms have little practical significance.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- Більшість сучасних СУБД базуються на реляційній моделі даних, однак усі ці реалізації володіють суттєвими відмінностями від цієї теоретичної моделі. У практичному варіанті реляційної моделі відношення називають *таблицями*, атрибути – *колонками*, а кортежі – *рядками*.
- Most modern databases are based on a relational data model, but all these implementations have significant differences from this theoretical model. In the practical version of the relational model, relations are called *tables*, attributes are called *columns*, and tuples are called *rows*.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- У практичній реалізації реляційної моделі допускається використання невизначених значень атрибутів, які зазвичай позначаються як NULL. Передбачається, що невизначене значення може задаватися, коли значення атрибута не відоме, не визначене або не має сенсу у поєднанні зі значеннями інших атрибутів того ж рядку.
- In the practical implementation of the relational model, the use of indeterminate attribute values, which are usually denoted as NULL, is allowed. It is assumed that an undefined value can be specified when the value of an attribute is unknown, undefined, or meaningless in combination with the values of other attributes of the same row.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- Використання невизначених значень спрощує проектування схеми БД за рахунок деякого ускладнення коду програм. При відображенні у БД ієрархії класів усі об'єкти можуть відображені у одну таблицю; атрибути, присутні тільки у об'єктів підкласу, отримають невизначені значення для об'єктів суперкласу. Приналежність об'єкту до певного класу повинна встановлюватися у кодї, щоб запобігти некоректне використання атрибутів NULL.
- The use of undefined values simplifies the design of the database schema due to some complication of the program code. When displaying a class hierarchy in a database, all objects can be mapped to a single table; attributes present only in subclass objects will be given undefined values for superclass objects. The object must belong to a certain class in the code to prevent the NULL attributes from being used incorrectly.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- Досить часто невизначені значення використовуються у випадках, коли значення не є обов'язковим. Наприклад, для пасажирів, який не має карти постійного клієнта, значення відповідного атрибуту може бути невизначеним. У теоретичній реляційній моделі невизначені значення не допускаються.
- Quite often undefined values are used in cases where the value is not mandatory. For example, for a passenger who does not have a regular customer card, the value of the corresponding attribute may be uncertain. In the theoretical relational model, indefinite values are not allowed.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- Використання невизначених значень суттєво ускладнює основні операції і призводить до появи парадоксів. Наприклад, операції над атрибутами зі значеннями NULL у результаті дають значення NULL. Однак операція фільтрації інтерпретує невизначене значення логічного предикату як хибне.
- The use of indefinite values significantly complicates basic operations and leads to paradoxes. For example, operations on attributes with NULL values result in NULL values. However, the filtering operation interprets the indefinite value of the logical predicate as false.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- При проектуванні баз даних доброю практикою вважається заборона на використання невизначених значень для усіх атрибутів, окрім тих, для яких необхідна підтримка невизначених значень. Заборона на використання невизначених значень є обмеженням цілісності.
- When designing databases, it is good practice to prohibit the use of undefined values for all attributes except those that require the support of undefined values. Prohibiting the use of undefined values is an integrity constraint.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- На відміну від теоретичної реляційної моделі практичні реалізації допускають появу ідентичних рядків як у таблицях, так і у результатах виконання запитів. Використання дублікатів дозволяється, оскільки усунення або перевірка дублікатів є обчислювально складними операціями та могли призводити до значного зниження продуктивності системи.
- In contrast to the theoretical relational model, practical implementations allow the appearance of identical rows in both tables and query results. The use of duplicates is allowed because eliminating or verifying duplicates is a computationally complex operation and could lead to a significant reduction in system performance.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- У сучасних умовах вплив обчислень, необхідних для видалення дублікатів, на загальну продуктивність систем став значно меншим внаслідок зростання потужності обчислювальних систем у декілька разів. Наявність дублікатів у таблицях частіше за все є наслідком помилки розробників програм.
- In modern conditions, the impact of computing required to remove duplicates on the overall performance of systems has become much smaller due to the increase in the capacity of computing systems several times. The presence of duplicates in the tables is most often the result of a mistake by software developers.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- Можливість використання невизначених значень та допущення дублікатів призводять до зміни семантики та алгебраїчних властивостей реляційних операцій у СУБД. Наприклад, допущення дублікатів призводить до необхідності розрізняти теоретико-множинні операції, що виключають дублікати з результату та залишають їх.
- The possibility of using undefined values and the assumption of duplicates lead to changes in the semantics and algebraic properties of relational operations in the DBMS. For example, the assumption of duplicates leads to the need to distinguish between set-theoretic operations that exclude duplicates from the result and leave them.

4.6 Реляційна модель даних. Практичні аспекти

4.6 Relational data model. Practical aspects

- Практичні реалізації також передбачають операції зовнішнього з'єднання, результати яких можуть містити невизначені значення. Наприклад, операція лівого зовнішнього з'єднання включає до результату ті кортежі першого аргументу, для яких не знайшлось пари у другому аргументі, доповнюючи ці кортежі невизначеними значеннями для атрибутів другого аргументу.
- Practical implementations also involve external connection operations, the results of which may contain uncertain values. For example, the left external join operation includes in the result those tuples of the first argument for which no pair was found in the second argument, supplementing these tuples with undefined values for the attributes of the second argument.

Контрольні питання

1. Основні поняття реляційної моделі даних.
2. Реляційна алгебра.
3. Реляційне обчислення.
4. Функціональні залежності та ключі.
5. Нормалізація та нормальні форми.
6. Практичні аспекти застосування реляційної моделі даних.

Assessment questions

1. Basic concepts of relational data model.
2. Relational algebra.
3. Relational calculus.
4. Functional dependencies and keys.
5. Normalization and normal forms.
6. Practical aspects of application of relational data model.

187

Моделювання даних Data modeling

Тема 5
Торіс 5

188

5 Моделювання даних 5 Data modeling

- Проектування інформаційних систем і баз даних, що знаходяться у їх основі, є достатньо складною задачею. Опис усіх використаних у системі або взаємодіючих з нею типів об'єктів реального світу може нараховувати сотні або тисячі одиниць, співвідношення та зв'язки між якими частіше за все є нетривіальними та потребують глибоких знань процесів.
- Designing the information systems and databases that underlie them is quite a challenge. The description of all types of real-world objects used in the system or interacting with it can number hundreds or thousands of units, the relationships and connections between which are often non-trivial and require in-depth knowledge of the processes.

5 Моделювання даних

5 Data modeling

- Для спрощення порозуміння між спеціалістами предметної галузі та розробниками інформаційних систем використовуються різні мови моделювання, серед яких широко відома уніфікована мова об'єктного моделювання UML та модель даних концептуального рівня “сутність-зв'язок” (Entity-Relationship, ER).
- Various modeling languages are used to facilitate communication between subject area experts and information system developers, including the widely known object modeling language UML and the Entity-Relationship (ER) conceptual data model.

5 Моделювання даних

5 Data modeling

- Різниця між цими інструментами моделювання у тому, що UML дозволяє описати високорівневу об'єктну модель усієї системи, включаючи поведінку тощо, тоді як ER здебільшого описує властивості даних, які використовуються у системі. Існують й інші засоби моделювання, які описують окремі сторони функціонування системи, наприклад мови моделювання бізнес-процесів.
- The difference between these modeling tools is that UML allows you to describe a high-level object model of the entire system, including behavior, etc., while ER mostly describes the properties of the data used in the system. There are other modeling tools that describe certain aspects of the system, such as business process modeling languages.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Одним з основних понять моделі є поняття *сутності*. Сутність є описом деякого об'єкту реального світу, який може бути чітко відокремлений від інших об'єктів (також представлених іншими сутностями моделі), а його опис однозначно пов'язаний з цим реальним об'єктом.
- One of the basic concepts of the model is the concept of *entity*. An entity is a description of an object in the real world that can be clearly separated from other objects (also represented by other entities in the model), and its description is uniquely related to that real object.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Сутності повинні розрізнятись між собою. Спочатку визначається наявність деякого способу ідентифікації, що дозволяє зіставити об'єкти реального світу з їх представленнями у БД. Передбачається ідентифікація сутностей за їх природними признаками, а не сурогатними ідентифікаторами.
- Entities must be different. At first, you need to determine the presence of some method of identification, which allows you to compare real-world objects with their representations in the database. The identification of entities by their natural features, rather than surrogate identifiers, is considered.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Необхідно щоб описи сутностей у моделі були пов'язані з об'єктами реального світу. Значення атрибутів сутностей можуть змінюватися, але повинен бути деякий незмінюваний ідентифікатор. Передбачається, що усі атрибути мають скалярні значення, для ідентифікації атрибутів використовують імена.
- The descriptions of the entities in the model need to be related to real-world objects. The values of entity attributes can vary, but there must be some immutable identifier. It is assumed that all attributes have scalar values, using names to identify attributes.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Сукупність сутностей, що мають однакові (за іменами) набори атрибутів, називається *множиною сутностей*. Це поняття є аналогом класу у об'єктних моделях. Крім того множина сутностей можуть називатися сутностями, а їх елементи – екземплярами сутностей.
- A collection of entities that have the same (by name) attribute sets is called a *set of entities*. This concept is analogous to the class in object models. In addition, a set of entities can be called entities, and their elements – instances of entities.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Наприклад, у системі зберігається інформація про виробників та моделі автомобілів та літаків. За значенням атрибуту “виробник” можна визначити, що Ford – автомобіль, а Airbus – літак. Проте цей набір атрибутів може описувати будь-які транспортні засоби. Тому для введення відмінностей між об'єктами необхідно включити нові атрибути, специфічні для автомобілів та літаків.
- For example, the system stores information about manufacturers and models of cars and aircrafts. The value of the "manufacturer" attribute can determine that Ford is a car and Airbus is an airplane. However, this set of attributes can describe any vehicle. Therefore, to introduce differences between objects, you need to include new attributes specific to cars and airplanes.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Інший приклад – у системі необхідно зберігати дані про постачальників та споживачів продукції підприємства. Скоріш за все, постачальники та споживачі будуть мати ті ж самі атрибути. Одна й та ж сутність може бути як постачальником, так і споживачем, тому об'єднання усіх партнерів у єдиній множині сутностей цілком виправдане.
- Another example – the system must store data about suppliers and consumers of enterprise products. Most likely, suppliers and consumers will have the same attributes. The same entity can be both a supplier and a consumer, so combining all the partners into a single set of entities is justified.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Іншим ключовим поняттям моделі є *зв'язок*. Зв'язком є упорядкована послідовність сутностей, що має свою ідентифікацію та може мати власні атрибути. Зв'язки між сутностями (з одних й тих самих множин) складають множину зв'язків.
- Another key concept of the model is *relationship*. A relationship is an ordered sequence of entities that has its own identification and may have its own attributes. Relationships between entities (from the same sets) make up a set of relationships.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Наприклад, сутності “студент” та “дисципліна” можуть бути пов'язані зв'язком “екзамен”. Набір сутностей (студент, дисципліна) буде ідентифікувати зв'язок, а додатковими атрибутами зв'язку можуть бути оцінка та дата її отримання.
- For example, the entities "student" and "discipline" may be related to the "exam" relationship. A set of entities (student, discipline) will identify the relationship, and additional attributes of the relationship may be a grade and the date of its receipt.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model



5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Найбільш поширеним видом зв'язків є бінарні (що поєднують дві сутності). Для них можна визначити обмеження щодо того як саме сутності можуть бути пов'язані: 1 – у зв'язку бере участь одна сутність; 0 – у зв'язку беруть участь не більше 1 сутності; m, n – у зв'язку бере участь 0 або декілька сутностей з однієї множини.
- The most common type of relationship is binary (combining two entities). For them, you can define restrictions on how the entities can be related: 1 – one entity is involved in the relationship; 0 – no more than 1 entity is involved in the relationship; m, n – 0 or more entities from one set are involved in the relationship.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Обмеження цілісності описуються парою символів, розділених двокрапкою. Наприклад, якщо співробітник обов'язково належить до одного відділу, таке обмеження можна виразити як 1:n (*один-до-багатьох*); n позначає, що декілька співробітників можуть бути пов'язані з одним відділом, а 1 – що кожний співробітник обов'язково пов'язаний лише з одним відділом.
- Integrity constraints are described by a pair of characters separated by a colon. For example, if an employee must belong to the same department, this restriction can be expressed as 1: n (*one-to-many*); n indicates that multiple employees can be associated with one department, and 1 indicates that each employee must be associated with only one department.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- З іншого боку, зв'язок між викладачем та науковим ступенем підлягає обмеженню 0:n, оскільки викладач не обов'язково повинен мати науковий ступінь. Найбільш складним типом зв'язку є m:n (*багато-до-багатьох*). Наприклад, співробітник може брати участь у декількох проектах, в кожному з яких беруть участь декілька співробітників.
- On the other hand, the relationship between a teacher and a PhD degree is limited to 0:n, as the teacher does not have to have a PhD degree. The most complex type of connection is m:n (*many-to-many*). For example, an employee may be involved in several projects, each involving multiple employees.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Об'єкти реального світу можуть одночасно входити до складу декількох множин. Людина може бути (або не бути) співробітником деякого підприємства. Такі ситуації представляються на основі *спадкування* – бінарного зв'язку 1:0 (*один-до-одного*). При спадкуванні атрибути більш широкої сутності не дублюються у більш вузькій сутності.
- Real-world objects can be part of several sets at the same time. A person may or may not be an employee of an enterprise. Such situations are represented on the basis of *inheritance* – binary relationship 1:0 (*one-to-one*). When inheriting, the attributes of a more general entity are not duplicated in a more specific entity.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

Побудовану модель “сутність-зв'язок” необхідно відобразити у реляційну модель даних наступним чином:

1. Для кожної множини сутностей побудувати відношення, атрибутами якого будуть усі атрибути сутностей, що входять до цієї множини.
2. Визначити функціональні залежності атрибутів від ідентифікаторів для кожного відношення.
3. Для кожної множини зв'язків побудувати відношення, атрибутами якого будуть усі атрибути зв'язку. Ідентифікатор зв'язку буде містити усі ідентифікатори пов'язаних сутностей.
4. Визначити функціональні залежності атрибутів, отриманих з атрибутів зв'язку, від атрибутів, отриманих з ідентифікатора зв'язку.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

The designed entity-relationship model must be mapped to a relational data model as following:

1. For each set of entities build a relation which attributes will be all the attributes of the entities included in this set.
2. Determine the functional dependencies of the attributes on the identifiers for each relation.
3. For each set of relationships, construct a relation which attributes will be all the attributes of the relationship. The relationship identifier will contain all the identifiers of the related entities.
4. Determine the functional dependencies of the attributes derived from the relationship attributes on the attributes derived from the relationship identifiers.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Якщо інші функціональні залежності визначити не можливо, то отримані відношення будуть у 3NF. Інакше отримана схема не буде нормалізованою та, можливо, знадобиться додаткова нормалізація. Це може вказувати на помилки проектування моделі “сутність-зв'язок” (можливо, деякі сутності насправді є агрегатами, які складаються з інших сутностей).
- If other functional dependencies cannot be determined, then the obtained relations will be in 3NF. Otherwise, the resulting scheme will be non-normalized and may require additional normalization. This may indicate errors in the design of the entity-relationship model (perhaps some entities are actually aggregates that consist of other entities).

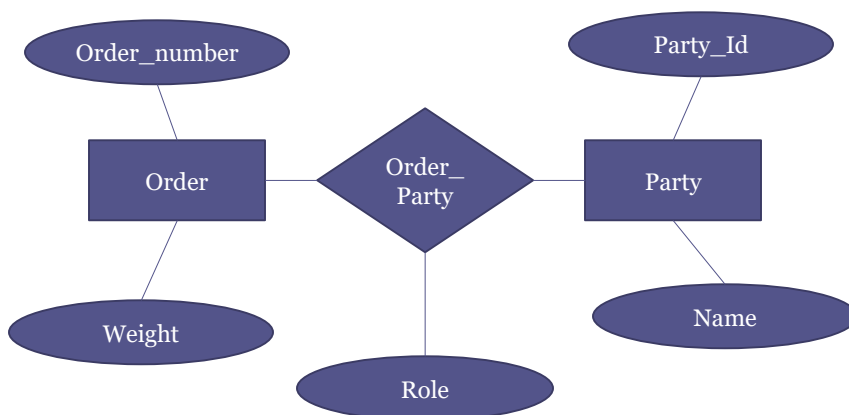
5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Наприклад, з кожним замовленням на перевезення пов'язані відправник та отримувач. Кожна особа може бути як відправником, так і отримувачем, тому їх можна поєднати у сутність party. Атрибутом зв'язку між замовленням order та особою буде роль особи у цьому зв'язку, а також ідентифікатори пов'язаних сутностей.
- For example, each order for transportation is associated with the sender and the recipient. Each person can be both a sender and a recipient, so they can be combined in the entity called party. The attribute of the relationship between the order and the party will be the role of the party in this relationship, as well as the identifiers of the related entities.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model



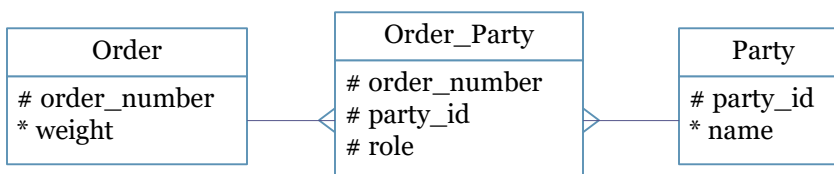
5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Ключами у відношеннях *order* та *party* будуть *order_number* та *party_id* відповідно, а ключем відношення *order_party* буде сукупність усіх його атрибутів, оскільки одна особа може виступати по відношенню до вантажу у декількох ролях одночасно.
- The keys in the *order* and *party* relations will be *order_number* and *party_id*, respectively, and the key in the *order_party* relation will be the set of all its attributes, as one person can have several roles simultaneously.

5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model



5.1 Моделювання даних. Модель сутність-зв'язок

5.1 Data modeling. Entity-relationship model

- Для побудови діаграм моделі даних “сутність-зв'язок” не передбачено можливості визначення зв'язків $m:n$, тому для представлення таких зв'язків необхідно визначати додаткові сутності та два зв'язки типу $1:n$. Ключ іншого відношення, який функціонально залежить від ключа відношення та включений до складу його атрибутів, називається *зовнішнім ключем*.
- It is not possible to define $m:n$ relationships to build entity-relationship data model diagrams, so you must define additional entities and two $1:n$ relationships to represent such relations. The key of another relationship, which functionally depends on the key of the relationship and is included in its attributes, is called a *foreign key*.

5.2 Моделювання даних. Концептуальна модель

5.2 Data modeling. Conceptual model

- Сучасні варіанти моделі “сутність-зв'язок” надають велику кількість різних можливостей для опису структур даних та особливостей їх взаємозв'язків, але не дозволяють описувати будь-які операції над цими структурами даних.
- Modern variants of the entity-relationship model provide a large number of different possibilities for describing data structures and features of their relationships, but do not allow to describe any operations on these data structures.

5.2 Моделювання даних. Концептуальна модель

5.2 Data modeling. Conceptual model

- Зазвичай ця особливість моделі розглядається як недолік, який вирішується в рамках об'єктно-орієнтованого підходу до проектування систем. В даний час домінуючим засобом об'єктно-орієнтованого проектування є уніфікована мова моделювання UML.
- This feature of the model is usually seen as a shortcoming that is solved through an object-oriented approach to system design. Currently, the dominant tool of object-oriented design is a unified modeling language UML.

5.2 Моделювання даних. Концептуальна модель

5.2 Data modeling. Conceptual model

- У процесі моделювання з використанням UML одночасно створюються діаграма класів програми та відповідна до неї модель даних. Зазвичай такі схеми БД поступаються якістю схемам, отриманим при використанні моделі “сутність-зв'язок”. Це викликано необхідністю компромісів для спрощення відображення схеми БД у структури даних програми.
- In the process of modeling using UML, a class diagram and the corresponding data model are created at the same time. Typically, such database schemas have weak quality in compare to schemas obtained using the entity-relationship model. This is due to the need for compromises to simplify the mapping of the database schema into the data structure of the program.

5.3 Моделювання даних. Об'єктна модель

5.3 Data modeling. Object model

Очікувалось, що об'єктні БД витіснять усі інші класи моделей даних, однак цього не сталося через:

- БД може використовуватися лише клієнтами, написаними на одній мові програмування;
- Не вдалося створити високорівневу декларативну мову запитів;
- Не вдалося забезпечити високу продуктивність при масовій обробці даних.

Object databases were expected to displace all other classes of data models, but this did not happen due to:

- The database can be used only by clients written in one programming language;
- Failed to create high-level declarative query language;
- It was not possible to ensure high performance in mass data processing.

5.3 Моделювання даних. Об'єктна модель

5.3 Data modeling. Object model

- Системи, засновані на суто об'єктних моделях даних, зайняли відносно невеликий сегмент, але деякі об'єктні засоби стали складовою частиною СУБД загального призначення. Зараз усі системи реалізують об'єктні розширення, тому їх прийнято називати об'єктно-реляційними.
- Systems based on purely object data models occupied a relatively small segment, but some object tools became part of the general-purpose database. Now all systems implement object extensions, so they are called object-relational.

5.3 Моделювання даних. Об'єктна модель

5.3 Data modeling. Object model

Найбільш важливими видами об'єктних розширень вважаються:

- Можливість визначення користувацьких типів даних, у тому числі й структурних;
- Використання колекцій об'єктів;
- Можливість створення структурних типів у доповнення до скалярних.

The most important types of object extensions are:

- Ability to define custom data types, including structural;
- Using object collections;
- Ability to create structural types in addition to scalar.

5.3 Моделювання даних. Об'єктна модель

5.3 Data modeling. Object model

Колекція – набір об'єктів визначеного типу.

Розрізняють наступні види колекцій:

- Set – набір об'єктів, що не містить дублікатів;
- Bag – набір об'єктів, що може містити дублікати;
- List – впорядкований список об'єктів;
- Array – набір об'єктів з доступом за індексом.

A collection is a set of objects of a certain type.

There are the following types of collections:

- Set – a set of objects that does not contain duplicates;
- Bag – a set of objects that can contain duplicates;
- List – an ordered list of objects;
- Array – a set of objects with index access.

5.4 Моделювання даних. Слабкоструктуровані моделі

5.4 Data modeling. Semi-structured models

- У деяких системах відокремлення опису структури даних (схеми) від самих даних не є бажаним. У таких випадках значна частина даних представляє собою текст на природній мові, тому такі дані називають слабкоструктурованими. Частіше за все це дані у форматах XML та JSON.
- In some systems, separating the description of the data structure (scheme) from the data itself is not desirable. In such cases, a significant part of the data is a text in natural language, so such data are called semi-structured. Most often it is data in XML and JSON formats.

5.4 Моделювання даних. Слабкоструктуровані моделі

5.4 Data modeling. Semi-structured models

- Хоча XML та JSON не передбачались для зберігання даних, ці формати можна розглядати як моделі даних. Вони надають можливості для більш гнучкого опису та представлення даних, ніж реляційна модель, однак схеми XML (XSD) надають засоби опису не менш жорстких обмежень ніж реляційна модель.
- Although XML and JSON were not intended for data storage, these formats can be considered as data models. They provide opportunities for more flexible description and representation of data than a relational model, but XML (XSD) schemas provide tools for describing no less strict constraints than a relational model.

5.4 Моделювання даних. Слабкоструктуровані моделі

5.4 Data modeling. Semi-structured models

- У об'єктно-реляційних БД слабкоструктуровані дані можуть зберігатися як значення атрибутів таблиць. Для цього застосовуються типи даних xml, json тощо. Вбудовані функції дозволяють формувати значення слабкоструктурованих типів з табличних даних та, навпаки, вилучати елементи слабкоструктурованих значень у вигляді колекцій.
- In object-relational databases, poorly structured data can be stored as values of table attributes; xml, json, etc. data types are used for this purpose. The built-in functions allow to form values of semi-structured types from tabular data and, vise versa, to extract elements of semi-structured values in the form of collections.

5.5 Моделювання даних. Моделі подання знань

5.5 Data modeling. Knowledge-representation models

- У системах представлення знань (семантичні мережі та онтології) та для представлення графів часто використовується тернарна модель даних, у якій елементарною структурою є трійка вигляду (об'єкт, атрибут, значення). Це дає дуже велику гнучкість представлення будь-яких об'єктів, проте значно впливає на складність обробки та продуктивність.
- Knowledge representation systems (semantic networks and ontologies) and graph representations often use a ternary data model, in which the elementary structure is a triple of views (object, attribute, value). This gives a lot of flexibility to the presentation of any object, but significantly affects the complexity of processing and productivity.

5.6 Моделювання даних. Моделі ключ-значення

5.6 Data modeling. Key-value models

- У останні роки широко відомими стали системи зберігання пар ключ-значення. Передбачається, що пошук даних можливий лише за ключем, а інтерпретація значення виконується у програмі. Однак усі функції СУБД (взаємозв'язки між об'єктами, обмеження цілісності, мови запитів, пошук за не ключовими атрибутами) повинні бути реалізовані у програмі.
- In recent years, key-value pair storage systems have become widely known. It is assumed that the search for data is possible only by key, and the interpretation of the value is performed in the program. However, all DBMS functions (object relationships, integrity constraints, query languages, non-key attribute searches) must be implemented in the program.

5.7 Моделювання даних. Застарілі моделі даних

5.7 Data modeling. Outdated data models

- Найбільш відомими серед застарілих на сьогодні моделей даних є мережева та ієрархічна моделі. Вони надавали можливості навігаційного доступу до окремих об'єктів та опису зв'язків між ними, однак у ієрархічній моделі представлення даних завжди є деревом. Можливості мережевої моделі повністю перекриті об'єктно-реляційними моделями, а ієрархічної – засобами JSON та XML.
- The most well-known among the outdated data models today are network and hierarchical models. They provided navigational access to individual objects and a description of the relationships between them, but in a hierarchical model, data representation is always a tree. The capabilities of the network model are completely overlaid with object-relational models, and the hierarchical model is completely overlaid with JSON and XML tools.

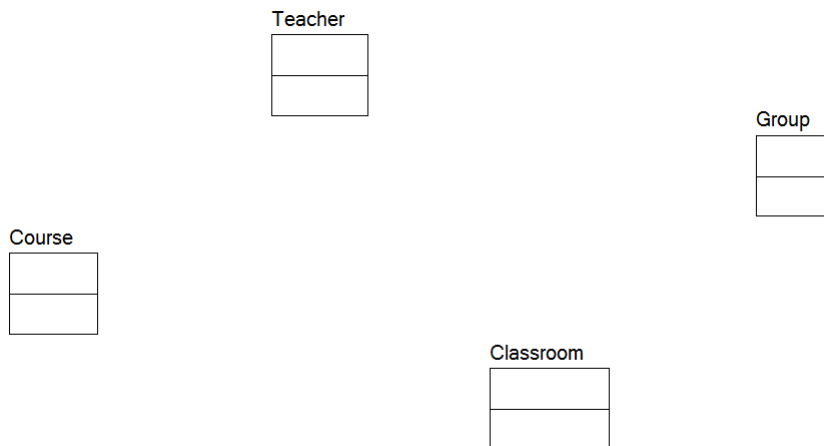
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples

- Розглянемо схему бази даних, призначеної для зберігання інформації про розклад навчальних занять. Цей фрагмент значно простіший за реальні схеми, проте навіть така спрощена схема дає можливість показати використання різних типів зв'язків.
- Consider the scheme of the database designed to store information about the schedule of classes. This fragment is much simpler than real schemes, but even such a simplified scheme allows to show the use of different types of connections.

5.8 Моделювання даних. Приклади моделювання БД

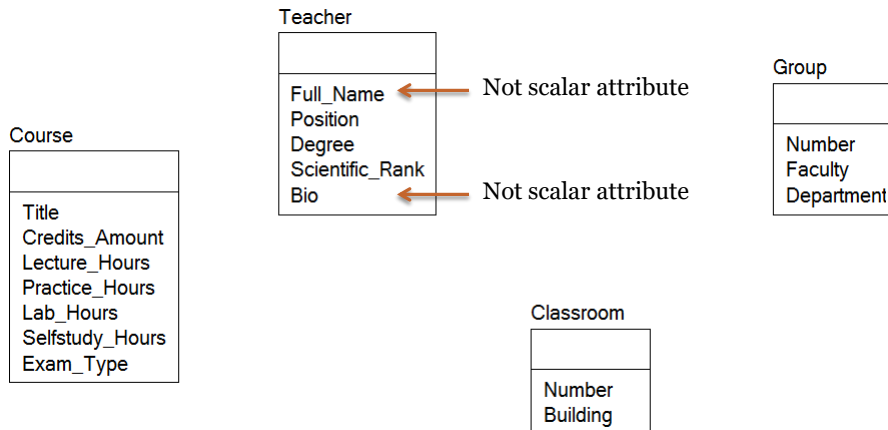
5.8 Data modeling. DB modeling examples



227

5.8 Моделювання даних. Приклади моделювання БД

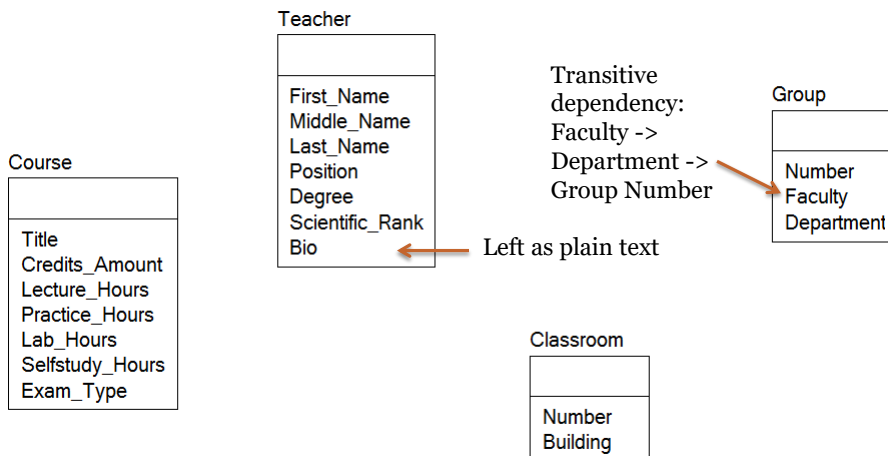
5.8 Data modeling. DB modeling examples



228

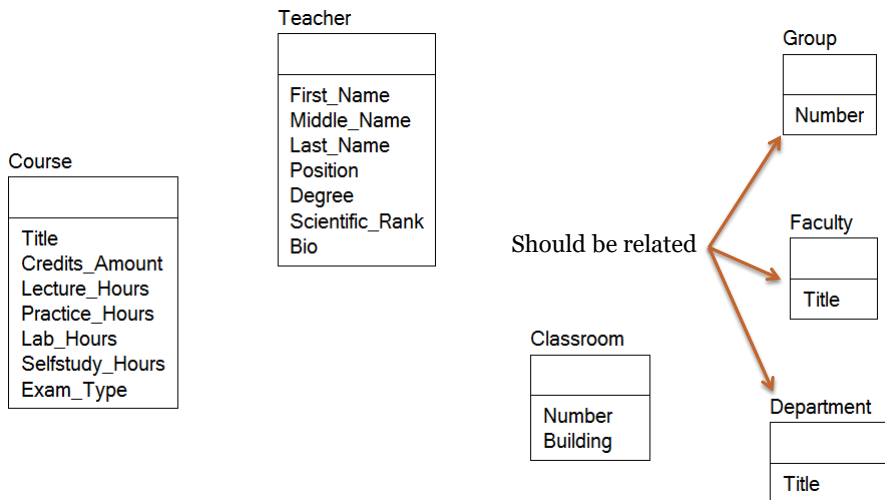
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



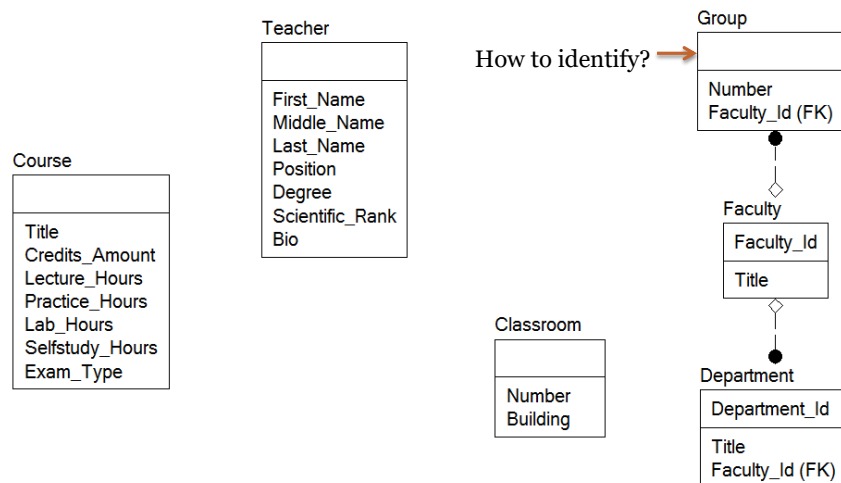
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



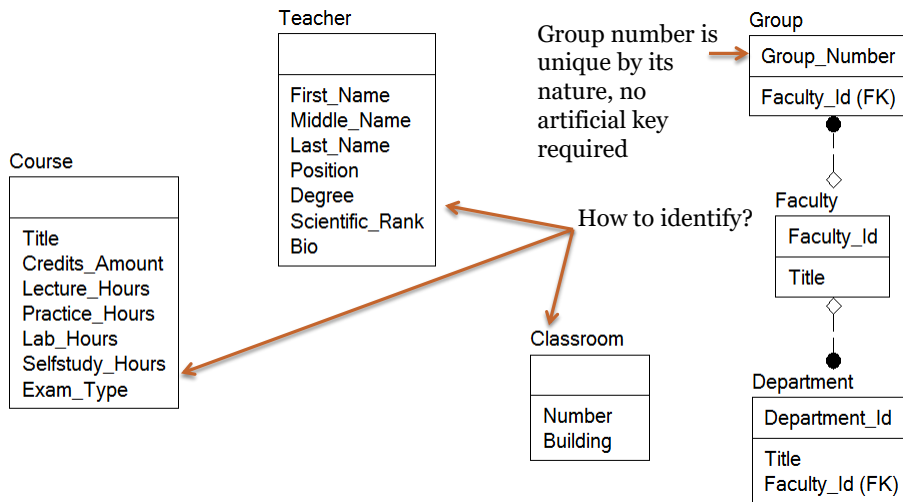
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



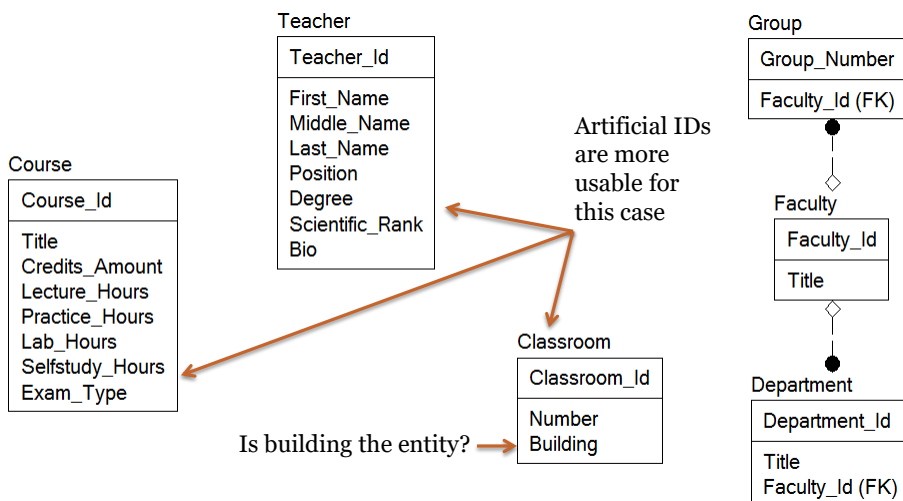
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



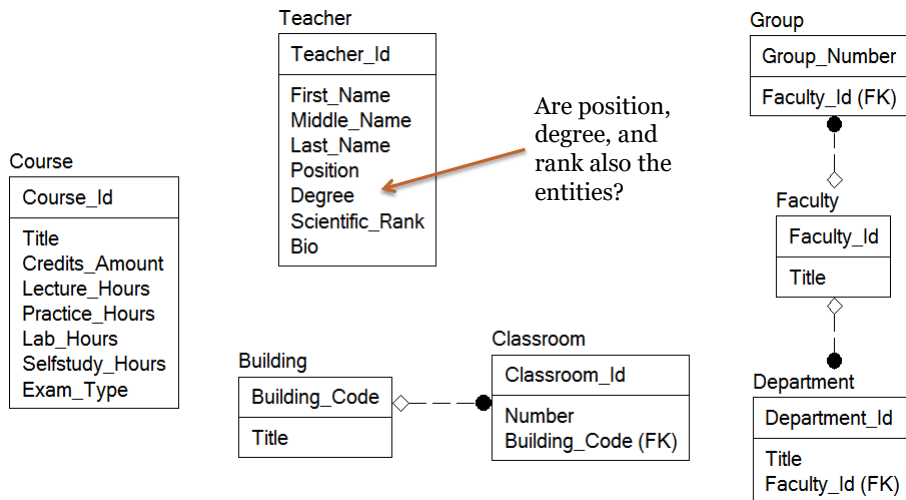
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



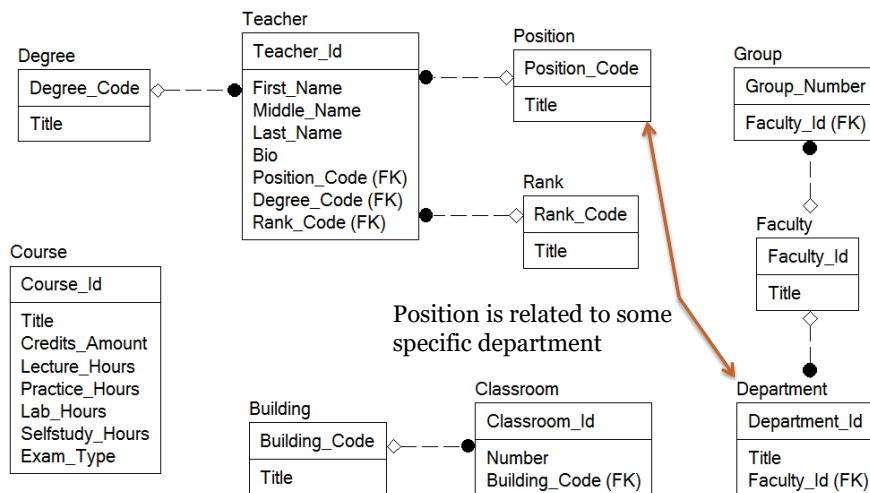
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



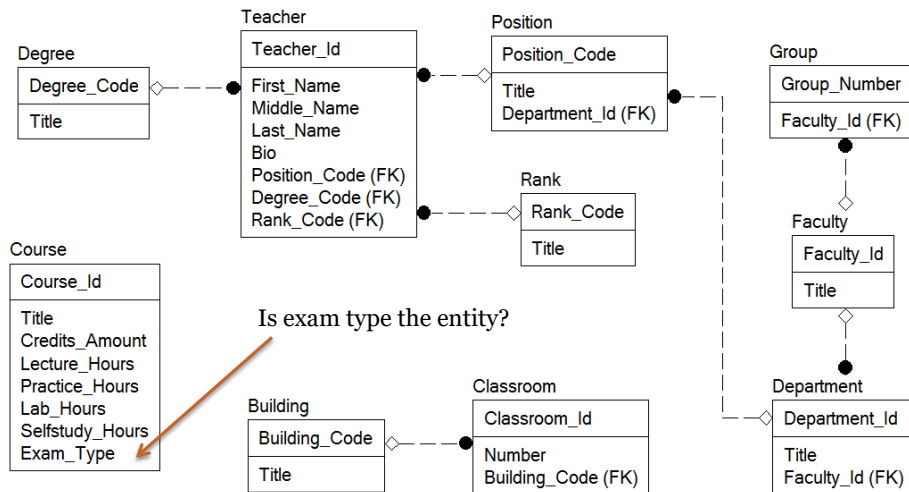
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



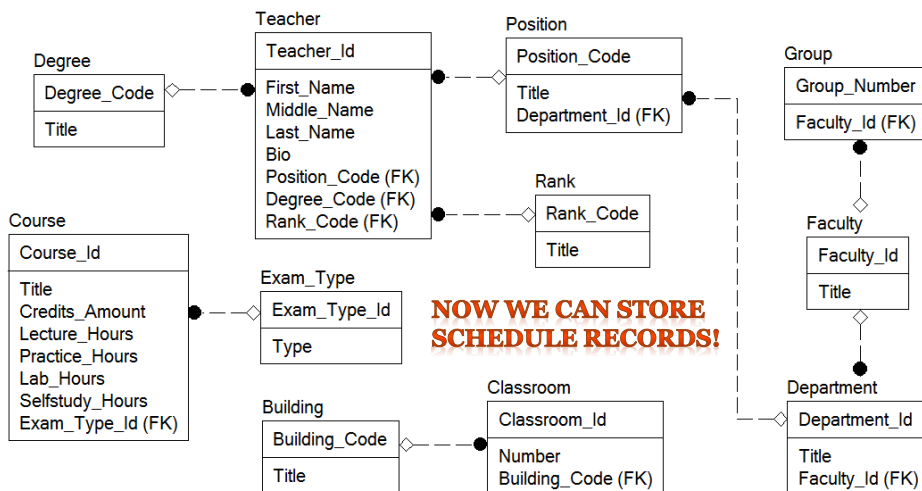
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



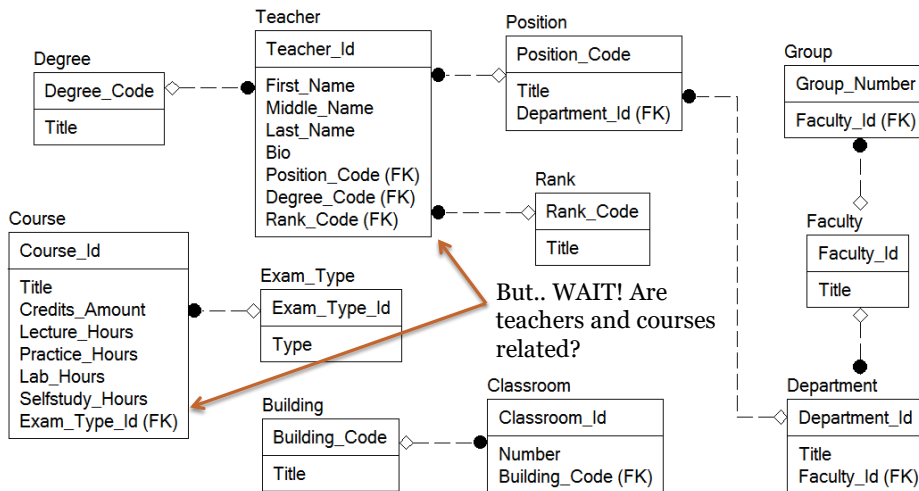
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



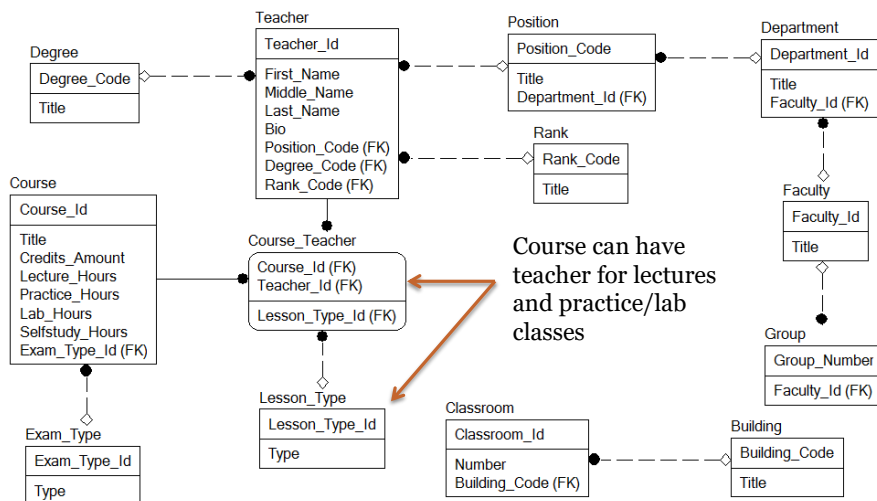
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



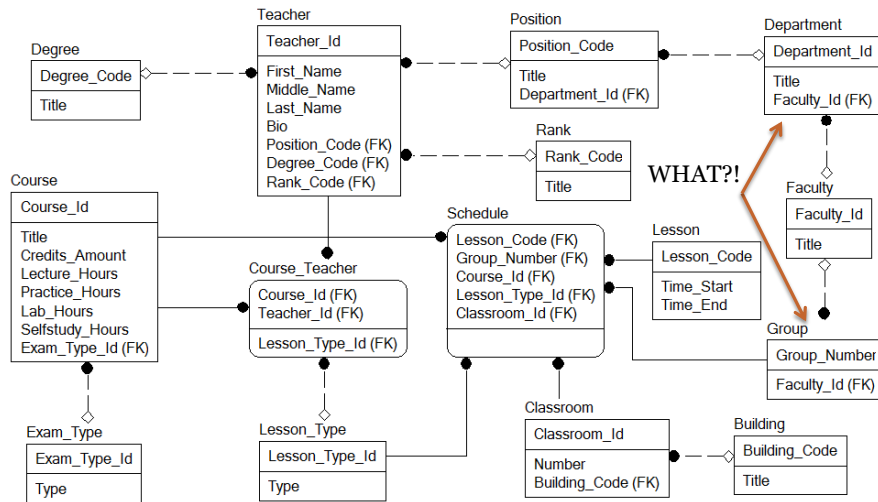
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



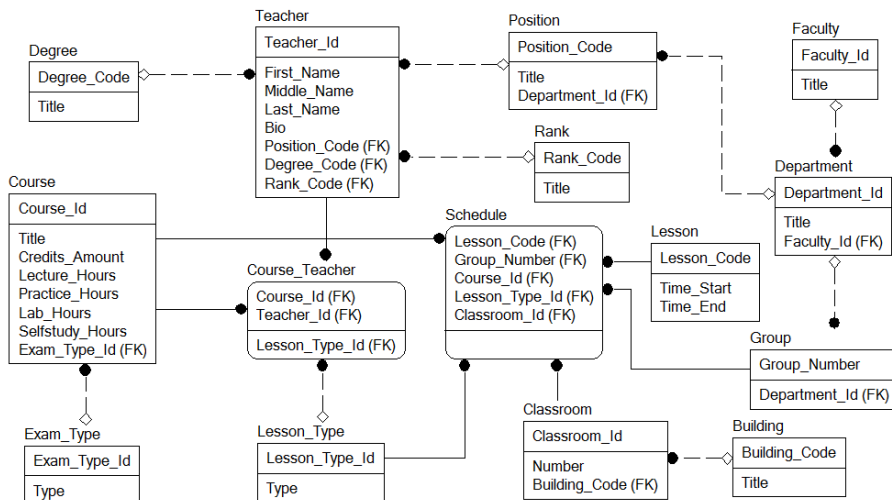
5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



5.8 Моделювання даних. Приклади моделювання БД

5.8 Data modeling. DB modeling examples



241

Контрольні питання

1. Модель сутність-зв'язок.
2. Об'єктна та інші моделі даних.
3. Проектування схеми БД у моделі сутність-зв'язок.

242

Assessment questions

1. The entity-relationship model.
2. Object and other data models.
3. Designing a database schema in an entity-relationship model.

Мова SQL SQL Language

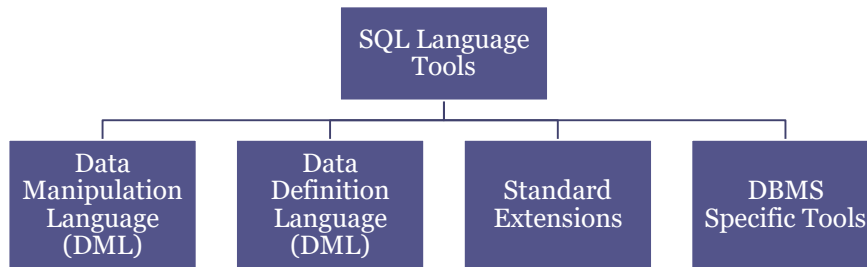
Тема 6
Topic 6

6.1 Призначення мови SQL 6.1 Purpose of the SQL language

- Однією з основних вимог до СУБД є наявність високорівневих засобів виконання запитів. У реляційних СУБД таким засобом є мова SQL. Фактично ця мова містить повний набір операцій, необхідних для виконання будь-яких дій з БД.
- One of the main requirements for the database is the availability of high-level tools of query execution. In relational databases, such a tool is the SQL language. In fact, this language contains a complete set of operations required to perform any action on the database.

6.1 Призначення мови SQL

6.1 Purpose of the SQL language



6.1 Призначення мови SQL

6.1 Purpose of the SQL language

- **Засоби маніпулювання даними** (Data Manipulation Language, DML) забезпечують виконання пошуку, вилучення, додавання, зміни та видалення даних, визначених у описі логічної структури бази даних, але не дозволяють змінювати цю структуру.
- **Data Manipulation Language** (DML) tools allow search, extract, add, modify, and delete data defined in the description of the logical structure of the database, but do not allow you to change this structure.

6.1 Призначення мови SQL

6.1 Purpose of the SQL language

- **Засоби визначення даних** (Data Definition Language, DDL) забезпечують створення, модифікацію та видалення елементів опису структури бази даних – як логічної, так і структури зберігання.
- **Data Definition Language** (DDL) tools allow create, modify, and delete database structure description elements, of both logical and physical structure.

6.1 Призначення мови SQL

6.1 Purpose of the SQL language

- **Розширення**, передбачені стандартом, але не включені у основне ядро мови SQL.
- **Додаткові засоби**, що забезпечують опис особливостей, специфічних для конкретної СУБД. Частіше за все це – визначення конкретних структур зберігання або їх параметрів.
- **Extensions** provided by the standard, but not included in the main core of the SQL language.
- **Additional tools** that provide a description of features specific to a particular database. Most often it is the definition of specific storage structures or their parameters.

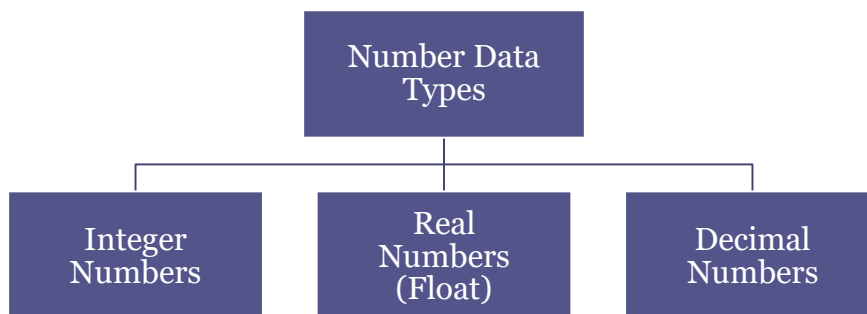
6.2 Прості типи даних

6.2 Simple data types

- У реляційній моделі даних значення атрибутів є елементами доменів. Домени описують властивості деяких конкретних типів значень. Однак у SQL не була врахована специфіка доменів та допускалось використання тільки таких типів, як числа, текстові рядки, моменти часу та деякі інші, не прив'язані до конкретних доменів.
- In a relational data model, attribute values are domain elements. Domains describe the properties of some specific value types. However, SQL did not take into account the specifics of domains and allowed the use of only such types as numbers, text strings, time points, and some others that are not tied to specific domains.

6.2 Прості типи даних

6.2 Simple data types



251

6.2 Прості типи даних

6.2 Simple data types

- **Цілі числа** (int, integer) зазвичай представляють собою двійкові числа, для яких визначені алгебраїчні операції та операції порівняння.
- **Integers** (int, integer) are usually binary numbers for which algebraic operations and comparison operations are defined.

252

6.2 Прості типи даних

6.2 Simple data types

- **Дійсні числа** (real, double precision) також зазвичай зберігаються у внутрішньому представленні комп'ютера, на якому працює СУБД, та можуть мати різну точність. Звичайно для них також визначені усі звичайні операції та відношення.
- **Real numbers** (real, double precision) are also usually stored in the internal representation of the computer on which the database is deployed, and can have different accuracy. Of course, all the usual operations and relationships are also defined for them.

6.2 Прості типи даних

6.2 Simple data types

- **Десяткові числа** (decimal, numeric) обробляються за правилами десятичної арифметики з фіксованим положенням десятичної точки (яка відокремлює дробову частину від цілої). Для цих типів можна вказувати максимальну кількість цифр та кількість цифр після десятичної точки.
- **Decimal numbers** (decimal, numeric) are processed according to the rules of decimal arithmetic with a fixed position of the decimal point (which separates the fractional part from the integer). For these types, you can specify the maximum number of digits and the number of digits after the decimal point.

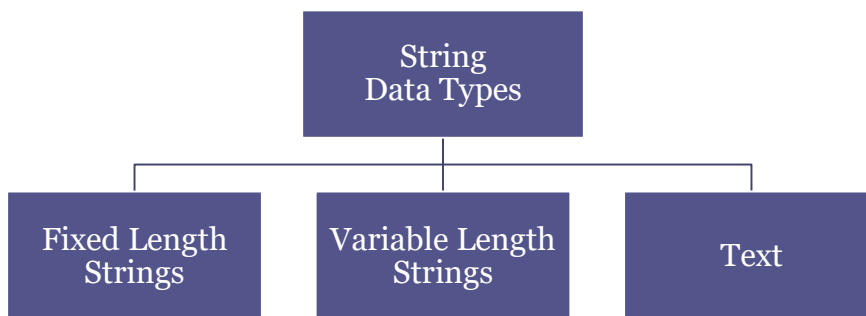
6.2 Прості типи даних

6.2 Simple data types

- Десяткові числа корисні для зберігання грошових значень. У багатьох країнах, у яких основна грошова одиниця складається зі 100 більш дрібних одиниць, законодавство вимагає виконання усіх обчислень зі збереженням трьох знаків після десятичної крапки (до 0.1 цента або копійки).
- Decimal numbers are useful for storing monetary values. In many countries, where the basic currency consists of 100 smaller units, the law requires all calculations to be performed with three decimal places (up to 0.1 cent or kopeck).

6.2 Прості типи даних

6.2 Simple data types



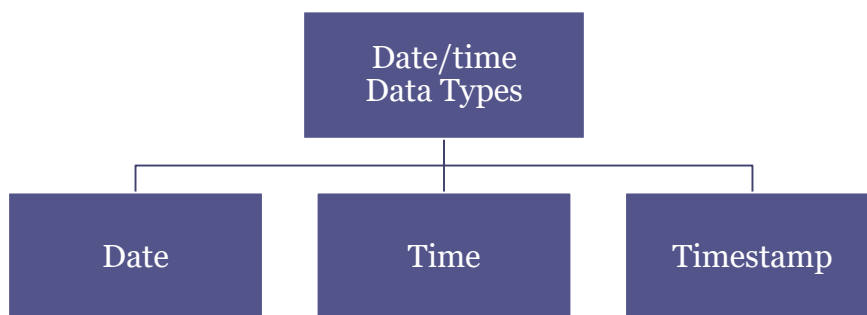
6.2 Прості типи даних

6.2 Simple data types

- Для зберігання символічних даних можна використовувати **рядки фіксованої довжини** (char) та **рядки змінної довжини** (varchar, text). Тип varchar вимагає вказувати максимальну довжину, а text – ні та придатний для зберігання рядків великого розміру. У багатьох СУБД text рекомендується використовувати завжди, коли обмеження максимальної довжини не потребується.
- You can use **fixed-length** (char) and **variable-length** (varchar, text) **strings** to store character data. The *varchar* type requires a maximum length, but *text* does not and is suitable for storing large strings. In many DBMS *text* it is recommended to use whenever the maximum length restriction is not required.

6.2 Прості типи даних

6.2 Simple data types



6.2 Прості типи даних

6.2 Simple data types

- Для часу передбачені типи **дат** (*date*), **часу доби** (*time*) та позначок часу (*timestamp*), які включають і дату, і час. Для типів, що містять час (*time* та *timestamp*), існують різновиди, що включають (або не включають) інформацію про часовий пояс.
- For time points, the types of **date** (*date*), **time of day** (*time*), and **timestamps** (*timestamp*), which include both date and time, are provided. For types that contain time (*time* and *timestamp*), there are implementations that include (or do not include) time zone information.

6.2 Прості типи даних

6.2 Simple data types

- **Логічний тип** (boolean), визначений у багатьох СУБД, як і слід було очікувати, зберігає булеві значення (істина або брехня).
- The **boolean type** defined in many DBMSs, as expected, stores boolean values (true or false).

6.3 SQL

Intro

- SQL став стандартом Американського національного інституту стандартів (ANSI) у 1986 році та Міжнародної організації зі стандартизації (ISO) у 1987 році.
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.

6.3 SQL

Intro

Для чого використовується SQL?

- SQL може виконувати запити до бази даних
- SQL може отримувати дані з бази даних
- SQL може вставляти записи в базу даних
- SQL може оновлювати записи в базі даних
- SQL може видаляти записи з бази даних
- SQL може створювати нові бази даних
- SQL може створювати нові таблиці в базі даних
- SQL може створювати збережені процедури в базі даних
- SQL може створювати подання в базі даних
- SQL може встановлювати дозволи на таблиці, процедури та подання

6.3 SQL

Intro

What can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

6.3 SQL

Intro

- Хоча SQL є стандартом ANSI/ISO, існують різні версії мови SQL. Однак, щоб відповідати стандарту ANSI, усі вони підтримують принаймні основні команди (наприклад, SELECT, UPDATE, DELETE, INSERT, WHERE) подібним чином. Більшість СУБД також мають свої власні розширення на додаток до стандарту SQL.
- Although SQL is an ANSI/ISO standard, there are different versions of the SQL language. However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner. Most of the DBMS also have their own proprietary extensions in addition to the SQL standard.

6.3 SQL

Intro

Щоб створити веб-сайт, який відображає дані з бази даних, вам знадобиться:

- СУБД для роботи з базою даних (тобто MS SQL Server, MySQL);
- Використовувати серверну мову сценаріїв, наприклад PHP;
- Використовувати SQL для отримання потрібних даних;
- Використовувати HTML/CSS для стилізації сторінки.

To build a web site that shows data from a database, you will need:

- An RDBMS database program (i.e. MS SQL Server, MySQL);
- To use a server-side scripting language, like PHP;
- To use SQL to get the data you want;
- To use HTML/CSS to style the page.

6.3 SQL Syntax

- Ключові слова SQL НЕ чутливі до регістру: select – це те саме, що SELECT. Деякі системи баз даних вимагають крапки з комою в кінці кожного оператора SQL. Крапка з комою – це стандартний спосіб відокремлення кожного оператора SQL у системах баз даних, які дозволяють виконувати більше одного оператора SQL в одному і тому ж запиті до сервера.
- SQL keywords are NOT case sensitive: select is the same as SELECT. Some database systems require a semicolon at the end of each SQL statement. Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

6.3 SQL Syntax

SELECT – вилучає дані з бази даних
 UPDATE – оновлення даних у базі даних
 DELETE – видаляє дані з бази даних
 INSERT INTO – вставляє нові дані в базу даних
 CREATE DATABASE – створює нову базу даних
 ALTER DATABASE – змінює базу даних
 CREATE TABLE – створює нову таблицю
 ALTER TABLE – змінює таблицю
 DROP TABLE – видаляє таблицю
 CREATE INDEX – створює індекс (ключ пошуку)
 DROP INDEX – видаляє індекс

6.3 SQL Syntax

SELECT – extracts data from a database
 UPDATE – updates data in a database
 DELETE – deletes data from a database
 INSERT INTO – inserts new data into a database
 CREATE DATABASE – creates a new database
 ALTER DATABASE – modifies a database
 CREATE TABLE – creates a new table
 ALTER TABLE – modifies a table
 DROP TABLE – deletes a table
 CREATE INDEX – creates an index (search key)
 DROP INDEX – deletes an index

6.3 SQL SELECT

- Оператор SELECT використовується для вибору даних із бази даних. Повернуті дані зберігаються в таблиці результатів, яка називається набором результатів.
- The SELECT statement is used to select data from a database. The data returned is stored in a result table, called the result-set.

6.3 SQL SELECT

```
SELECT column1, column2, ...
FROM table_name;
```

- Тут стовпець1, стовпець2, ... – це імена полів таблиці, з якої потрібно вибрати дані. Якщо ви хочете вибрати всі поля, доступні в таблиці, використовуйте такий синтаксис:
- Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

6.3 SQL SELECT

Customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL SELECT

- Наступний оператор SQL вибирає стовпці "CustomerName" та "City" з таблиці "Customers":
- The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

```
SELECT CustomerName, City FROM Customers;
```

- Наступний оператор SQL вибирає всі стовпці з таблиці "Customers":
- The following SQL statement selects all the columns from the "Customers" table:

```
SELECT * FROM Customers;
```

```
SELECT CustomerName, City FROM Customers;
```

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.
Around the Horn	London
Berglunds snabbköp	Luleå

```
SELECT * FROM Customers;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL SELECT DISTINCT

- Оператор SELECT DISTINCT використовується для повернення лише різних значень. У середині таблиці стовпець часто містить багато повторюваних значень; а іноді потрібно лише перерахувати різні значення.
- The SELECT DISTINCT statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

6.3 SQL SELECT DISTINCT

- Наступний оператор SQL вибирає ВСІ (включаючи повторювані) значення зі стовпця "Country" в таблиці "Customers":
- The following SQL statement selects ALL (including the duplicates) values from the "Country" column in the "Customers" table:

```
SELECT Country FROM Customers;
```

- Наступний оператор SQL вибирає лише значення DISTINCT зі стовпця "Country" в таблиці "Customers":
- The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

```
SELECT DISTINCT Country FROM Customers;
```

275

```
SELECT Country FROM Customers;
```

Country
Germany
Mexico
Mexico
UK
Sweden

```
SELECT DISTINCT Country FROM Customers;
```

Country
Germany
Mexico
Sweden
UK

276

6.3 SQL WHERE

- WHERE використовується для фільтрації записів. WHERE використовується для вилучення лише тих записів, які відповідають заданій умові. WHERE використовується не тільки в операторі SELECT, воно також використовується в операторах UPDATE та DELETE.
- The WHERE clause is used to filter records. The WHERE clause is used to extract only those records that fulfill a specified condition. The WHERE clause is not only used in SELECT statement, it is also used in UPDATE, DELETE statement.

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

6.3 SQL WHERE

- Наступний оператор SQL вибирає всіх клієнтів з країни "Mexico" в таблиці "Customers":
- The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

```
SELECT * FROM Customers WHERE Country = 'Mexico';
```

- SQL вимагає одинарних лапок навколо текстових значень (більшість систем баз даних також дозволяють подвійні лапки). Однак числові поля не повинні міститися в лапках:
- SQL requires single quotes around text values (most database systems will also allow double quotes). However, numeric fields should not be enclosed in quotes:

```
SELECT * FROM Customers WHERE CustomerID = 1;
```

SELECT * FROM Customers WHERE Country = 'Mexico';						
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

SELECT * FROM Customers WHERE CustomerID = 1;						
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

6.3 SQL WHERE

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

6.3 SQL AND, OR, NOT

WHERE можна поєднувати з операторами AND, OR та NOT. Оператори AND та OR використовуються для фільтрації записів на основі більш ніж однієї умови:

- Оператор AND відображає запис, якщо всі умови, відокремлені AND, є TRUE.
- Оператор OR відображає запис, якщо будь-яка з умов, розділених OR – TRUE.
- Оператор NOT відображає запис, якщо умова(и) NOT TRUE.

The WHERE clause can be combined with AND, OR, and NOT operators. The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.
- The NOT operator displays a record if the condition(s) is NOT TRUE.

6.3 SQL AND, OR, NOT

- AND

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

- OR

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

- NOT

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

Customers						
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Leblans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
17	Drachenblut Delikatessend	Sven Ottlieb	Walsertweg 21	Aachen	52066	Germany
18	Du monde entier	Janine Labrune	67, rue des Cinquante Otages	Nantes	44000	France
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
22	FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	C/ Moralzarzal, 86	Madrid	28034	Spain
23	Foies gourmandes	Martine Rancé	184, chaussée de Tournai	Lille	59000	France
24	Folk och få HB	Maria Larsson	Åkergatan 24	Bräcke	S-844 67	Sweden
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

Record: 1 of 91 | No Filter | Search

6.3 SQL AND, OR, NOT

- Наступний оператор SQL вибирає всі поля з "Customers", де країна не "Germany" та не "USA":
- The following SQL statement selects all fields from "Customers" where country is NOT "Germany" and NOT "USA":

SELECT * FROM Customers

WHERE NOT Country = 'Germany' AND NOT Country = 'USA';

Query1							
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country	
32	GROSELLA-Restaurante	Manuel Pereira	5ª Ave. Los Palos Grandes	Caracas	1081	Venezuela	
34	Hanari Carnes	Mario Pontes	Rua do Paço, 67	Rio de Janeiro	05454-876	Brazil	
35	HILARIÓN-Abastos	Carlos Hernández	Carrera 22 con Ave. Carlos Soublette	San Cristóbal	5022	Venezuela	
37	Hungry Owl All-Night Grocers	Patricia McKenna	8 Johnstown Road	Cork		Ireland	
38	Island Trading	Helen Bennett	Garden House Crowther Way	Cowes	PO31 7PJ	UK	
40	La corne d'abondance	Daniel Tonini	67, avenue de l'Europe	Versailles	78000	France	
41	La maison d'Asie	Annette Roulet	1 rue Alsace-Lorraine	Toulouse	31000	France	
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada	
46	LILA-Supermercado	Carlos González	Carrera 52 con Ave. Bolívar #65-10	Barquisimeto	3508	Venezuela	
47	LINO-Delicateses	Felipe Izquierdo	Ave. 5 de Mayo Porlamar	I. de Margarita	4980	Venezuela	
49	Magazzini Alimentari Riuniti	Giovanni Rovelli	Via Ludovico il Moro 22	Bergamo	24100	Italy	
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium	
51	Mère Paillarde	Jean Fresnière	43 rue St. Laurent	Montréal	H1J 1C3	Canada	
53	North/South	Simon Crowther	South House 300 Queensbridge	London	SW7 1RZ	UK	
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Pí	Buenos Aires	1010	Argentina	
57	Paris spécialités	Marie Bertrand	265, boulevard Charonne	Paris	75012	France	
58	Pericles Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cash 321	México D.F.	05033	Mexico	
59	Piccolo und mehr	Georg Pippas	Geislsberg 14	Salzburg	5020	Austria	
60	Princesa Isabel Vinhos	Isabel de Castro	Estrada da saúde n. 58	Lisboa	1756	Portugal	
61	Que Delícia	Bernardo Batista	Rua da Panificadora, 12	Rio de Janeiro	02389-673	Brazil	
62	Queen Cozinha	Lúcia Carvalho	Alameda dos Canários, 891	São Paulo	05487-020	Brazil	
64	Rancho grande	Sergio Gutierrez	Av. del Libertador 900	Buenos Aires	1010	Argentina	
66	Reggiani Caseifici	Maurizio Moroni	Strada Provinciale 124	Reggio Emilia	42100	Italy	
67	Ricardo Adocicados	Janete Limeira	Av. Copacabana, 267	Rio de Janeiro	02389-890	Brazil	
68	Richter Supermarkt	Michael Holz	Grenzacherweg 237	Genève	1203	Switzerland	

Record: 1 of 67 | No Filter | Search

6.3 SQL ORDER BY

- Ключове слово ORDER BY використовується для сортування набору результатів за зростанням або спаданням. Ключове слово ORDER BY сортує записи за зростанням за замовчуванням. Для сортування записів за спаданням використовуйте ключове слово DESC.
- The ORDER BY keyword is used to sort the result-set in ascending or descending order. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC | DESC;
```

6.3 SQL ORDER BY

- Наступний оператор SQL вибирає всіх клієнтів з таблиці "Customers", відсортованих за стовпцем "Country":
- The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

```
SELECT * FROM Customers
ORDER BY Country;
```

297

Query1						
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
54	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Pis	Buenos Aires	1010	Argentina
59	Piccolo und mehr	Georg Pipps	Geislaweg 14	Salzburg	5020	Austria
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
81	Tradição Hipermercados	Anabela Domingues	Av. Inês de Castro, 414	São Paulo	05634-030	Brazil
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	08737-363	Brazil
31	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	04876-786	Brazil
62	Queen Cozinha	Lúcia Carvalho	Alameda dos Canários, 891	São Paulo	05487-020	Brazil
67	Ricardo Adocicados	Janete Limeira	Av. Copacabana, 267	Rio de Janeiro	02389-890	Brazil
61	Que Delícia	Bernardo Batista	Rua da Panificadora, 12	Rio de Janeiro	02389-673	Brazil
34	Hanari Carnes	Mario Pontes	Rua do Paço, 67	Rio de Janeiro	05454-876	Brazil
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
51	Mère Paillarde	Jean Fresnière	43 rue St. Laurent	Montréal	H1J 1C3	Canada
73	Simons bistro	Jytte Petersen	Vinbæltet 34	København	1734	Denmark
83	Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus	8200	Denmark
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	90110	Finland
74	Spécialités du monde	Dominique Perrier	25, rue Lauriston	Paris	75016	France
85	Vins et alcools Chevalier	Paul Henriot	59 rue de l'Abbaye	Reims	51100	France

Records: 14 of 91 No Filter Search

298

6.3 SQL ORDER BY

- Наступний оператор SQL вибирає всіх клієнтів з таблиці "Customers", відсортованих за зменшенням за стовпцем "Country":
- The following SQL statement selects all customers from the "Customers" table, sorted DESCENDING by the "Country" column:

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

299

Query1						
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
47	LINO-Delicatesses	Felipe Izquierdo	Ave. 5 de Mayo Porlamar	I. de Margarita	4980	Venezuela
33	GROSELLA-Restaurante	Manuel Pereira	5ª Ave. Los Palos Grandes	Caracas	1081	Venezuela
35	HILARIÓN-Abastos	Carlos Hernández	Carrera 22 con Ave. Carlos Soublette	San Cristóbal	5022	Venezuela
46	LILA-Supermercado	Carlos González	Carrera 52 con Ave. Bolívar #65-10	Barquisimeto	3508	Venezuela
82	Trail's Head Gourmet Provisioners	Helvetius Nagy	722 DaVinci Blvd.	Kirkland	98034	USA
77	The Big Cheese	Liz Nixon	89 Jefferson Way Suite 2	Portland	97201	USA
32	Great Lakes Food Market	Howard Snyder	2732 Baker Blvd.	Eugene	97403	USA
75	Split Rail Beer & Ale	Art Braunschweiger	P.O. Box 555	Lander	82520	USA
71	Save-a-Lot Markets	Jose Pavarotti	187 Suffolk Ln.	Boise	83720	USA
65	Rattlesnake Canyon Grocery	Paula Wilson	2817 Milton Dr.	Albuquerque	87110	USA
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
48	Lonesome Pine Restaurant	Fran Wilson	89 Chiaroscuro Rd.	Portland	97219	USA
78	The Cracker Box	Liu Wong	55 Grizzly Peak Rd.	Butte	59801	USA
45	Let's Stop N Shop	Jaime Yorres	87 Polk St. Suite 5	San Francisco	94117	USA
43	Lazy K Kountry Store	John Steel	12 Orchestra Terrace	Walla Walla	99362	USA
36	Hungry Coyote Import Store	Yoshi Latimer	City Center Plaza 516 Main St.	Elgin	97827	USA
55	Old World Delicatessen	Rene Phillips	2743 Bering St.	Anchorage	99508	USA
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
38	Island Trading	Helen Bennett	Garden House Crowther Way	Cowes	PO31 7PJ	UK
53	North/South	Simon Crowther	South House 300 Queensbridge	London	SW7 1RZ	UK
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
72	Seven Seas Imports	Hari Kumar	90 Wadhurst Rd.	London	OX15 4NB	UK
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
68	Richter Supermarket	Michael Holz	Grenzacherweg 237	Genève	1203	Switzerland

Record: 1 of 91 | No Filter | Search

300

6.3 SQL ORDER BY

- Наступний оператор SQL вибирає всіх клієнтів з таблиці "Customers", відсортованих за стовпцем "Country" та "CustomerName". Це означає, що він впорядковує за країною, але якщо деякі рядки мають однакову країну, він впорядковує їх за ім'ям клієнта:
- The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName:

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

301

Query1						
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Pis	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutierrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
31	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	04876-786	Brazil
34	Hanari Carnes	Mario Pontes	Rua do Paço, 67	Rio de Janeiro	05454-876	Brazil
61	Que Delícia	Bernardo Batista	Rua da Panificadora, 12	Rio de Janeiro	02389-673	Brazil
62	Queen Cozinha	Lúcia Carvalho	Alameda dos Canários, 891	São Paulo	05487-020	Brazil
67	Ricardo Adocicados	Janete Limeira	Av. Copacabana, 267	Rio de Janeiro	02389-890	Brazil
81	Tradição Hipermercados	Anabela Domingues	Av. Inês de Castro, 414	São Paulo	05634-030	Brazil
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	08737-363	Brazil
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
51	Mère Paillarde	Jean Fresnière	43 rue St. Laurent	Montréal	H1J 1C3	Canada
73	Simons bistro	Jytte Petersen	Vinbæltet 34	København	1734	Denmark
83	Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus	8200	Denmark
87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	90110	Finland
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France

Records: 14 | 1 of 91 | No Filter | Search

302

6.3 SQL ORDER BY

- Наступний оператор SQL вибирає всіх клієнтів з таблиці "Customers", відсортовані за зростанням за "Country" та за спаданням за стовпцем "CustomerName":
- The following SQL statement selects all customers from the "Customers" table, sorted ascending by the "Country" and descending by the "CustomerName" column:

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

303

Query1						
CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
54	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Pis	Buenos Aires	1010	Argentina
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	08737-363	Brazil
81	Tradição Hipermercados	Anabela Domingues	Av. Inês de Castro, 414	São Paulo	05634-030	Brazil
67	Ricardo Adocicados	Janete Limeira	Av. Copacabana, 267	Rio de Janeiro	02389-890	Brazil
62	Queen Cozinha	Lúcia Carvalho	Alameda dos Canários, 891	São Paulo	05487-020	Brazil
61	Que Delícia	Bernardo Batista	Rua da Panificadora, 12	Rio de Janeiro	02389-673	Brazil
34	Hanari Carnes	Mario Pontes	Rua do Paço, 67	Rio de Janeiro	05454-876	Brazil
31	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	04876-786	Brazil
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
51	Mère Pailarde	Jean Fresnière	43 rue St. Laurent	Montréal	H1J 1C3	Canada
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
83	Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus	8200	Denmark
73	Simons bistro	Jytte Petersen	Vinbæltet 34	København	1734	Denmark
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	90110	Finland
85	Vins et alcools Chevalier	Paul Henriot	59 rue de l'Abbaye	Reims	51100	France
84	Victuailles en stock	Mary Saveley	2, rue du Commerce	Lyon	69004	France

Record: 1 of 91 No Filter Search

304

6.3 SQL INSERT INTO

- Оператор INSERT INTO використовується для вставки нових записів у таблицю.
- The INSERT INTO statement is used to insert new records in a table.
- Написати оператор INSERT INTO можна двома способами. Перший спосіб визначає як імена стовпців, так і значення, які потрібно вставити:
- It is possible to write the INSERT INTO statement in two ways. The first way specifies both the column names and the values to be inserted:

6.3 SQL INSERT INTO

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

- Якщо ви додаєте значення для всіх стовпців таблиці, вам не потрібно вказувати імена стовпців у запиті SQL. Однак переконайтеся, що порядок значень має такий самий порядок, як і стовпці таблиці. Синтаксис INSERT INTO буде таким:
- If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. The INSERT INTO syntax would be as follows:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

6.3 SQL INSERT INTO

- Нижче наведено вибірку із таблиці "Customers" бази даних Northwind:
- Below is a selection from the "Customers" table in the Northwind sample database:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

6.3 SQL INSERT INTO

- Наступний оператор SQL вставляє новий запис у таблицю "Customers":
- The following SQL statement inserts a new record in the "Customers" table:

```
INSERT INTO Customers (CustomerName, ContactName,
    Address, City, PostalCode, Country)
VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen
    21', 'Stavanger', '4006', 'Norway');
```

6.3 SQL INSERT INTO

- Вибірка із таблиці "Customers" тепер виглядатиме так:
- The selection from the "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

6.3 SQL INSERT INTO

- Також можна вставляти дані лише у певні стовпці.

У наступному операторі SQL буде вставлено новий запис, але дані лише в стовпці "CustomerName", "City" та "Country" (Ідентифікатор клієнта буде оновлено автоматично):

- It is also possible to only insert data in specific columns.

The following SQL statement will insert a new record, but only insert data in the "CustomerName", "City", and "Country" columns (CustomerID will be updated automatically):

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

6.3 SQL INSERT INTO

- Вибірка із таблиці "Customers" тепер виглядатиме так:

- The selection from the "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	null	null	Stavanger	null	Norway

6.3 SQL NULL

- Поле зі значенням NULL – це поле без значення.

Якщо поле таблиці є необов'язковим, можна вставити новий запис або оновити запис, не додаючи значення в це поле. Потім поле буде збережено із значенням NULL.

- A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

6.3 SQL NULL

- Значення NULL відрізняється від нульового значення або поля, що містить пробіли. Поле зі значенням NULL – це поле, яке залишилось порожнім під час створення запису!
- A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

6.3 SQL NULL

- Неможливо перевірити значення NULL за допомогою операторів порівняння, таких як =, < або <>. Натомість нам доведеться використовувати оператори IS NULL та IS NOT NULL.
- It is not possible to test for NULL values with comparison operators, such as =, <, or <>. We will have to use the IS NULL and IS NOT NULL operators instead.

6.3 SQL NULL

IS NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;
```

IS NOT NULL Syntax

```
SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

6.3 SQL NULL

- Нижче наведено вибір із таблиці "Customers" БД Northwind:
- Below is a selection from the "Customers" table in the Northwind DB:

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL NULL

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```

Result ?

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NOT NULL;
```

Result ?

6.3 SQL UPDATE

- Оператор UPDATE використовується для модифікації існуючих записів у таблиці.
- The UPDATE statement is used to modify the existing records in a table.

UPDATE *table_name*
SET *column1* = *value1*, *column2* = *value2*, ...
WHERE *condition*;

6.3 SQL UPDATE

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL UPDATE

- Наступний вираз SQL оновлює першого клієнта (CustomerID = 1) новою контактною особою та новим містом.
- The following SQL statement updates the first customer (CustomerID = 1) with a new contact person and a new city.

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

6.3 SQL UPDATE

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL UPDATE

- Same WHERE визначає, скільки записів буде оновлено.
Наступний вираз SQL оновить ім'я контакту на "Juan" для всіх записів, де країною є "Mexico".
- It is the WHERE clause that determines how many records will be updated. The following SQL statement will update the contact name to "Juan" for all records where country is "Mexico".

```
UPDATE Customers
SET ContactName='Juan'
WHERE Country='Mexico';
```

6.3 SQL UPDATE

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

323

6.3 SQL UPDATE

- Будьте обережні під час оновлення записів. Якщо ви пропустите пункт WHERE, ВСІ записи будуть оновлені!
- Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

```
UPDATE Customers
SET ContactName='Juan';
```

324

6.3 SQL UPDATE

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Juan	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Juan	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Juan	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL DELETE

- Оператор DELETE використовується для видалення наявних записів у таблиці.
- The DELETE statement is used to delete existing records in a table.

DELETE FROM table_name WHERE condition;

6.3 SQL DELETE

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

327

6.3 SQL DELETE

- Наступний оператор SQL видаляє клієнта "Alfreds Futterkiste" із таблиці "Customers":
- The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

```
DELETE FROM Customers WHERE
  CustomerName='Alfreds Futterkiste';
```

328

6.3 SQL DELETE

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL DELETE

- Можна видалити всі рядки в таблиці, не видаляючи таблицю.
Це означає, що структура таблиці, атрибути та індекси будуть незмінними.
- It is possible to delete all rows in a table without deleting the table.
This means that the table structure, attributes, and indexes will be intact.

`DELETE FROM table_name;`

6.3 SQL DELETE

- Наступний оператор SQL видаляє всі рядки таблиці "Customers", не видаляючи таблицю.
- The following SQL statement deletes all rows in the "Customers" table, without deleting the table.

`DELETE FROM Customers;`

6.3 SQL SELECT TOP

- Конструкція SELECT TOP використовується для вказівки кількості записів, які потрібно повернути. Конструкція SELECT TOP корисна для великих таблиць з тисячами записів. Повернення великої кількості записів може вплинути на продуктивність.
- The SELECT TOP clause is used to specify the number of records to return. The SELECT TOP clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

6.3 SQL SELECT TOP

- Не всі системи баз даних підтримують вираз SELECT TOP. MySQL підтримує вираз LIMIT для вибору обмеженої кількості записів, тоді як Oracle використовує ROWNUM.
- Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records, while Oracle uses ROWNUM.

6.3 SQL SELECT TOP

Microsoft Access SQL Server	SELECT TOP number percent column_name(s) FROM table_name WHERE condition;
MySQL	SELECT column_name(s) FROM table_name WHERE condition LIMIT number;
Oracle	SELECT column_name(s) FROM table_name WHERE ROWNUM <= number;

6.3 SQL SELECT TOP

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

6.3 SQL SELECT TOP

- Наступний оператор SQL вибирає перші три записи з таблиці "Customers".
- The following SQL statement selects the first three records from the "Customers" table.

MS Access / SQL Server

```
SELECT TOP 3 * FROM Customers;
```

MySQL

```
SELECT * FROM Customers LIMIT 3;
```

Oracle

```
SELECT * FROM Customers WHERE ROWNUM <= 3;
```

6.3 SQL SELECT TOP

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

337

6.3 SQL SELECT TOP

- Наступний оператор SQL вибирає перші 50% записів із таблиці "Customers".
- The following SQL statement selects the first 50% of the records from the "Customers" table.

```
SELECT TOP 50 PERCENT * FROM Customers;
```

338

6.3 SQL SELECT TOP

- Наступний оператор SQL вибирає перші три записи з таблиці "Customers", де країною є "Germany".
- The following SQL statement selects the first three records from the "Customers" table, where the country is "Germany".

```
SELECT TOP 3 * FROM Customers  
WHERE Country='Germany';
```

6.3 SQL MIN, MAX

- Функція MIN () повертає найменше значення вибраного стовпця.

Функція MAX () повертає найбільше значення вибраного стовпця.

- The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

6.3 SQL MIN, MAX

MIN()

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX()

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

341

6.3 SQL MIN, MAX

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

342

6.3 SQL MIN, MAX

- Наступний SQL визначає ціну найдешевшого продукту:
- The following SQL finds the price of the cheapest product:

```
SELECT MIN(Price) AS SmallestPrice FROM Products;
```

- Наступний SQL визначає ціну найдорожчого продукту:
- The following SQL finds the price of the most expensive product:

```
SELECT MAX(Price) AS LargestPrice FROM Products;
```

6.3 SQL COUNT, AVG, SUM

- Функція COUNT () повертає кількість рядків, що відповідає заданому критерію.
 - Функція AVG () повертає середнє значення числового стовпця.
 - Функція SUM () повертає загальну суму числового стовпця.
- The COUNT() function returns the number of rows that matches a specified criterion.
 - The AVG() function returns the average value of a numeric column.
 - The SUM() function returns the total sum of a numeric column.

6.3 SQL COUNT, AVG, SUM

COUNT()	SELECT COUNT(<i>column_name</i>) FROM <i>table_name</i> WHERE <i>condition</i> ;
AVG()	SELECT AVG(<i>column_name</i>) FROM <i>table_name</i> WHERE <i>condition</i> ;
SUM()	SELECT SUM(<i>column_name</i>) FROM <i>table_name</i> WHERE <i>condition</i> ;

345

6.3 SQL COUNT, AVG, SUM

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

346

6.3 SQL COUNT, AVG, SUM

- Наступний оператор SQL знаходить кількість продуктів:
- The following SQL statement finds the number of products:

```
SELECT COUNT(ProductID)
FROM Products;
```

- Значення NULL не враховуються.
- NULL values are not counted.

347

6.3 SQL COUNT, AVG, SUM

- Наступний оператор SQL знаходить середню ціну всіх продуктів:
- The following SQL statement finds the average price of all products:

```
SELECT AVG(Price)  
FROM Products;
```

- Значення NULL ігноруються.
- NULL values are ignored.

348

6.3 SQL COUNT, AVG, SUM

- Наступний оператор SQL знаходить загальну ціну всіх продуктів:
- The following SQL statement finds the total price of all products:

```
SELECT SUM(Price)  
FROM Products;
```

- Значення NULL ігноруються.
- NULL values are ignored.

6.3 SQL LIKE

- Оператор LIKE використовується в операторі WHERE для пошуку вказаного шаблону в стовпці. У поєднанні з оператором LIKE часто використовуються два символи підстановки:
% - знак відсотка представляє нуль, один або кілька символів;
_ - підкреслення представляє один символ.
- The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. There are two wildcards often used in conjunction with the LIKE operator:
% - the percent sign represents zero, one, or multiple characters;
_ - the underscore represents a single character.

6.3 SQL LIKE

- MS Access використовує зірочку (*) замість знака відсотка (%), а знак запитання (?) замість підкреслення (_).
- MS Access uses an asterisk (*) instead of the percent sign (%), and a question mark (?) instead of the underscore (_).

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

6.3 SQL LIKE

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

6.3 SQL LIKE

- Наступний оператор SQL вибирає всіх клієнтів з CustomerName, що починається з "a":
- The following SQL statement selects all customers with a CustomerName starting with "a":

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';
```

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%a';
```



353

6.3 SQL LIKE

- Наступний оператор SQL вибирає всіх клієнтів з CustomerName, які мають "or" у будь-якому положенні:
- The following SQL statement selects all customers with a CustomerName that have "or" in any position:

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%or%';
```

354

6.3 SQL LIKE

- Наступний оператор SQL вибирає всіх клієнтів з CustomerName, які мають "r" на другій позиції:
- The following SQL statement selects all customers with a CustomerName that have "r" in the second position:

```
SELECT * FROM Customers
WHERE CustomerName LIKE '_r%';
```

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a__%';
```



???

6.3 SQL LIKE

- Наступний оператор SQL вибирає всіх клієнтів з ContactName, яке починається з "a" і закінчується "o":
- The following SQL statement selects all customers with a ContactName that starts with "a" and ends with "o":

```
SELECT * FROM Customers
WHERE ContactName LIKE 'a%o';
```

```
SELECT * FROM Customers
WHERE CustomerName NOT LIKE 'a%';
```



???

6.3 SQL IN

- Оператор IN дозволяє вказати кілька значень у реченні WHERE. Оператор IN – це скорочення декількох умов АБО.
- The IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions.

<pre>SELECT column_name(s) FROM table_name WHERE column_name IN (value1, value2, ...);</pre>	<pre>SELECT column_name(s) FROM table_name WHERE column_name IN (SELECT STATEMENT);</pre>
--	---

357

6.3 SQL IN

- Наступний оператор SQL вибирає всіх клієнтів, які перебувають у "Germany", "France" або "UK":
- The following SQL statement selects all customers that are located in "Germany", "France" or "UK":

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');
```

```
SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');
```



???

358

6.3 SQL IN

- Наступний оператор SQL вибирає всіх клієнтів з тих самих країн, що і постачальники:
- The following SQL statement selects all customers that are from the same countries as the suppliers:

```
SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```



Sub-query

6.3 SQL BETWEEN

- Оператор BETWEEN вибирає значення в межах заданого діапазону. Значення можуть бути числами, текстом або датами. Оператор BETWEEN включає: включені початкові та кінцеві значення.
- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

6.3 SQL BETWEEN

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	1	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	1	2	36 boxes	21.35

361

6.3 SQL BETWEEN

- Наступний оператор SQL вибирає всі продукти з ціною МІЖ 10 і 20:
- The following SQL statement selects all products with a price BETWEEN 10 and 20:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```



???

362

6.3 SQL BETWEEN

- Наступний оператор SQL вибирає всі продукти з ціною МІЖ 10 і 20. Крім того не показувати товари з CategoryID 1,2 або 3:
- The following SQL statement selects all products with a price BETWEEN 10 and 20. In addition do not show products with a CategoryID of 1,2, or 3:

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3);
```

363

6.3 SQL BETWEEN

- Наступний оператор SQL вибирає всі продукти з ProductName між Carnarvon Tigers та Mozzarella di Giovanni:
- The following SQL statement selects all products with a ProductName BETWEEN Carnarvon Tigers and Mozzarella di Giovanni:

```
SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon
    Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;
```

364

6.3 SQL BETWEEN

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	7/4/1996	3
10249	81	6	7/5/1996	1
10250	34	4	7/8/1996	2
10251	84	3	7/9/1996	1
10252	76	4	7/10/1996	2

6.3 SQL BETWEEN

- Наступний оператор SQL вибирає всі замовлення з OrderDate МІЖ '01-July-1996' та '31-July-1996':
- The following SQL statement selects all orders with an OrderDate BETWEEN '01-July-1996' and '31-July-1996':

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
```

6.3 SQL Aliases

- SQL-псевдоніми використовуються для надання таблиці або стовпцю таблиці тимчасового імені. Псевдоніми часто використовують, щоб зробити назви стовпців більш читабельними. Псевдонім існує лише на час запиту.
- SQL aliases are used to give a table, or a column in a table, a temporary name. Aliases are often used to make column names more readable. An alias only exists for the duration of the query.

367

6.3 SQL Aliases

Alias Column

```
SELECT column_name AS alias_name
FROM table_name;
```

Alias Table

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

368

6.3 SQL Aliases

Customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK

Orders

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10354	58	8	1996-11-14	3
10355	4	6	1996-11-15	1
10356	86	6	1996-11-18	2

369

6.3 SQL Aliases

Alias Column

```
SELECT CustomerName AS Customer,  
       ContactName AS [Contact Person]  
FROM Customers;
```

Alias Table

```
SELECT o.OrderID, o.OrderDate, c.CustomerName  
FROM Customers AS c, Orders AS o  
WHERE c.CustomerName = 'Around the Horn'  
      AND c.CustomerID = o.CustomerID;
```

370

6.3 SQL Joins

- Вираз JOIN використовується для поєднання рядків з двох або більше таблиць на основі відповідного стовпця між ними.
- A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

371

6.3 SQL Joins

Orders

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Customers

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

372

6.3 SQL Joins

- Зверніть увагу, що стовпець "CustomerID" в таблиці "Orders" посилається на "CustomerID" в таблиці "Customers". Зв'язок між двома наведеними вище таблицями встановлюється через стовпчик "CustomerID".
- Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

6.3 SQL Joins

- Тоді ми можемо записати наступний оператор SQL (який містить INNER JOIN), який вибирає записи, що мають відповідні значення в обох таблицях:
- Then, we can create the following SQL statement (that contains an INNER JOIN), that selects records that have matching values in both tables:

```
SELECT Orders.OrderID, Customers.CustomerName,
       Orders.OrderDate
FROM Orders INNER JOIN Customers ON
       Orders.CustomerID = Customers.CustomerID;
```

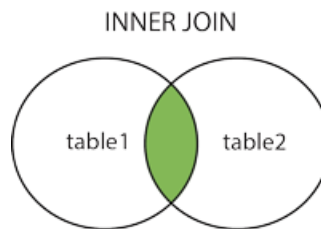
6.3 SQL Joins

```
SELECT Orders.OrderID, Customers.CustomerName,
       Orders.OrderDate
FROM Orders INNER JOIN Customers ON
       Orders.CustomerID = Customers.CustomerID;
```

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

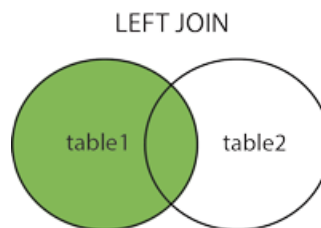
6.3 SQL Joins

- INNER JOIN: Повертає записи, що мають відповідні значення в обох таблицях
- INNER JOIN: Returns records that have matching values in both tables



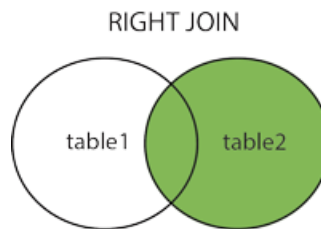
6.3 SQL Joins

- LEFT JOIN: Повертає всі записи з лівої таблиці та відповідні записи з правої таблиці
- LEFT JOIN: Returns all records from the left table, and the matched records from the right table



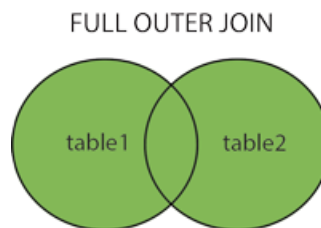
6.3 SQL Joins

- RIGHT JOIN: Повертає всі записи з правої таблиці та відповідні записи з лівої таблиці
- RIGHT JOIN: Returns all records from the right table, and the matched records from the left table

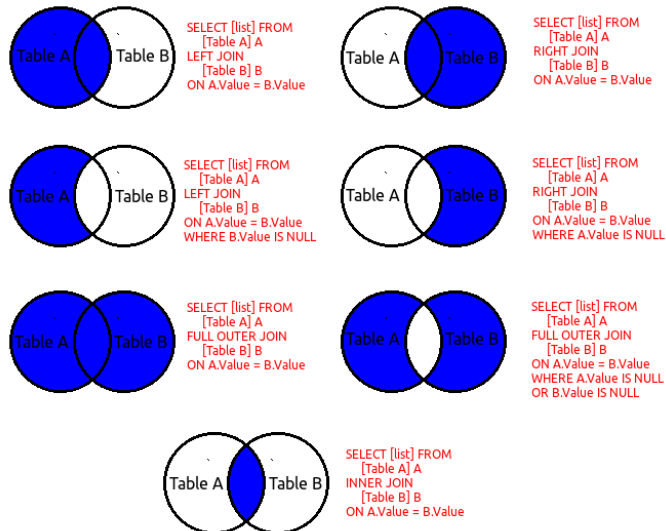


6.3 SQL Joins

- FULL OUTER JOIN: Повертає всі записи, якщо в лівій або правій таблиці є збіг
- FULL OUTER JOIN: Returns all records when there is a match in either left or right table



6.3 SQL Joins



6.3 SQL Joins

- Self JOIN – це звичайне об'єднання, але таблиця об'єднана сама з собою.
- A self JOIN is a regular join, but the table is joined with itself.

```
SELECT A.CustomerName AS CustomerName1,
       B.CustomerName AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID AND A.City = B.City
ORDER BY A.City;
```

6.3 SQL Joins

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Self JOIN

CustomerName1	CustomerName2	City
Ana Trujillo Emparedados y helados	Antonio Moreno Taquería	México D.F.

6.3 SQL UNION

Оператор UNION використовується для поєднання набору результатів двох або більше операторів SELECT.

- Кожен оператор SELECT у межах UNION повинен мати однакову кількість стовпців
- Стовпці також повинні мати подібні типи даних
- Стовпці в кожному операторі SELECT також повинні бути в однаковому порядку

```
SELECT column_name(s) FROM table1
UNION [ALL]
SELECT column_name(s) FROM table2;
```

Оператор UNION за замовчуванням вибирає лише різні значення. Щоб дозволити повторювані значення, використовуйте UNION ALL.

6.3 SQL UNION

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order

```
SELECT column_name(s) FROM table1
UNION [ALL]
SELECT column_name(s) FROM table2;
```

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL.

6.3 SQL UNION

Customers

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico

Suppliers

SupplierID	SupplierName	ContactName	Address	City	PostalCode	Country
1	Exotic Liquid	Charlotte Cooper	49 Gilbert St.	London	EC1 4SD	UK
2	New Orleans Cajun Delights	Shelley Burke	P.O. Box 78934	New Orleans	70117	USA
3	Grandma Kelly's Homestead	Regina Murphy	707 Oxford Rd.	Ann Arbor	48104	USA

385

6.3 SQL UNION

SELECT City FROM Customers
UNION
SELECT City FROM Suppliers
ORDER BY City;

DIFFERENCE ?

SELECT City FROM Customers
UNION ALL
SELECT City FROM Suppliers
ORDER BY City;

386

6.3 SQL UNION

SELECT City, Country FROM Customers
WHERE Country='Germany'

Get cities of German
customers only

UNION ALL

SELECT City, Country FROM Suppliers
WHERE Country='Germany'

Get cities of German
suppliers only

ORDER BY City;

Order German cities in
descending order

387

6.3 SQL UNION

```
SELECT 'Customer' AS Type, ContactName, City, Country FROM Customers
UNION
SELECT 'Supplier', ContactName, City, Country FROM Suppliers;
```

Merged set of
counterparties

Type	ContactName	City	Country
Customer	Yang Wang	Bern	Switzerland
Customer	Yoshi Latimer	Elgin	USA
Customer	Yoshi Tannamuri	Vancouver	Canada
Customer	Yvonne Moncada	Buenos Aires	Argentina
Customer	Zbyszek	Walla	Poland
Supplier	Anne Heikkonen	Lappeenranta	Finland
Supplier	Antonio del Valle Saavedra	Oviedo	Spain
Supplier	Beate Vileid	Sandvika	Norway
Supplier	Carlos Diaz	São Paulo	Brazil
Supplier	Chandra Leka	Singapore	Singapore

388

6.3 SQL GROUP BY

- Оператор GROUP BY групує рядки, що мають однакові значення, у зведені рядки, наприклад "знайти кількість клієнтів у кожній країні".
Оператор GROUP BY часто використовується з функціями агрегації (COUNT, MAX, MIN, SUM, AVG) для групування набору результатів за одним або кількома стовпцями.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

389

6.3 SQL GROUP BY

- The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".
The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

390

6.3 SQL GROUP BY

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

391

6.3 SQL GROUP BY

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

COUNT(CustomerID)	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
2	Denmark
2	Finland
11	France
11	Germany

392

6.3 SQL GROUP BY

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
ORDER BY COUNT(CustomerID) DESC;
```

COUNT(CustomerID)	Country
13	USA
11	Germany
11	France
9	Brazil
7	UK
5	Spain
5	Mexico
4	Venezuela

6.3 SQL HAVING

- Вираз HAVING було додано до SQL, оскільки ключове слово WHERE не можна використовувати з агрегатними функціями.
- The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions.

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

6.3 SQL HAVING

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

COUNT(CustomerID)	Country
9	Brazil
11	France
11	Germany
7	UK
13	USA

395

6.3 SQL HAVING

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 5
ORDER BY COUNT(CustomerID) DESC;
```

COUNT(CustomerID)	Country
13	USA
11	Germany
11	France
9	Brazil
7	UK

396

6.3 SQL HAVING

Orders

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	1996-07-04	3
10249	81	6	1996-07-05	1
10250	34	4	1996-07-08	2

Employees

EmployeeID	LastName	FirstName	BirthDate	Photo	Notes
1	Davolio	Nancy	1968-12-08	EmpID1.pic	Education includes a BA....
2	Fuller	Andrew	1952-02-19	EmpID2.pic	Andrew received his BTS....
3	Leverling	Janet	1963-08-30	EmpID3.pic	Janet has a BS degree....

397

6.3 SQL HAVING

```
SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders
FROM (Orders INNER JOIN Employees ON Orders.EmployeeID =
      Employees.EmployeeID)
GROUP BY LastName
HAVING COUNT(Orders.OrderID) > 10;
```

LastName	NumberOfOrders
Buchanan	11
Callahan	27
Davolio	29
Fuller	20
King	14
Leverling	31
Peacock	40
Suyama	18

398

6.3 SQL HAVING

```
SELECT Employees.LastName, COUNT(Orders.OrderID) AS Number
      OfOrders
FROM Orders INNER JOIN Employees ON Orders.EmployeeID =
      Employees.EmployeeID
WHERE LastName = 'Davolio' OR LastName = 'Fuller'
GROUP BY LastName
HAVING COUNT(Orders.OrderID) > 25;
```

LastName	NumberOfOrders
Davolio	29

6.3 SQL EXISTS

- Оператор EXISTS використовується для перевірки наявності будь-якого запису в підзапиті. Оператор EXISTS повертає true, якщо підзапит повертає один або кілька записів.
- The EXISTS operator is used to test for the existence of any record in a subquery. The EXISTS operator returns true if the subquery returns one or more records.

6.3 SQL EXISTS

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition);
```

Example 1

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.
SupplierID = Suppliers.supplierID AND Price < 20);
```

Example 2

```
SELECT SupplierName
FROM Suppliers
WHERE EXISTS (SELECT ProductName FROM Products WHERE Products.
SupplierID = Suppliers.supplierID AND Price = 22);
```


6.3 SQL ANY, ALL

- Оператори ANY та ALL використовуються з реченням WHERE або HAVING. Оператор ANY повертає true, якщо будь-яке із значень підзапиту відповідає умові. Оператор ALL повертає true, якщо всі значення підзапиту відповідають умові.
- The ANY and ALL operators are used with a WHERE or HAVING clause. The ANY operator returns true if any of the subquery values meet the condition. The ALL operator returns true if all of the subquery values meet the condition.

6.3 SQL ANY, ALL

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
(SELECT column_name FROM table_name WHERE condition);
```

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
(SELECT column_name FROM table_name WHERE condition);
```

The *operator* must be a standard comparison operator:
=, <>, !=, >, >=, <, <=

403

6.3 SQL ANY, ALL

Products

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

OrderDetails

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

404

6.3 SQL ANY, ALL

```
SELECT ProductName
FROM Products
WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails
  WHERE Quantity = 10);
```

```
SELECT ProductName
FROM Products
WHERE ProductID = ALL (SELECT ProductID FROM OrderDetails
  WHERE Quantity = 10);
```

6.3 SQL CASE

- Оператор CASE перевіряє умови і повертає значення, коли дотримано першу умову (як оператор IF-THEN-ELSE). Отже, як тільки умова є істинною, оператор поверне результат. Якщо жодні умови не відповідають дійсності, буде отримано значення у виразі ELSE. Якщо немає частини ELSE і жодні умови не відповідають дійсності, буде отримано NULL.

CASE

```
WHEN condition1 THEN result1
WHEN condition2 THEN result2
WHEN conditionN THEN resultN
ELSE result
```

END;

6.3 SQL CASE

- The CASE statement goes through conditions and returns a value when the first condition is met (like an IF-THEN-ELSE statement). So, once a condition is true, it will stop reading and return the result. If no conditions are true, it returns the value in the ELSE clause. If there is no ELSE part and no conditions are true, it returns NULL.

CASE

```
WHEN condition1 THEN result1
WHEN condition2 THEN result2
WHEN conditionN THEN resultN
ELSE result
```

END;

407

6.3 SQL CASE

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
3	10248	72	5
4	10249	14	9
5	10249	51	40

408

6.3 SQL CASE

```

SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;

```

OrderID	Quantity	QuantityText
10248	12	The quantity is under 30
10248	10	The quantity is under 30
10248	5	The quantity is under 30
10249	9	The quantity is under 30
10249	40	The quantity is greater than 30

6.3 SQL CASE

```
SELECT CustomerName, City, Country FROM Customers
ORDER BY
(CASE
  WHEN City IS NULL THEN Country
  ELSE City
END);
```

CustomerName	City	Country
Drachenblut Delikatessend	Aachen	Germany
Rattlesnake Canyon Grocery	Albuquerque	USA
Old World Delicatessen	Anchorage	USA
Vaffeljernet	Århus	Denmark
Galería del gastrónomo	Barcelona	Spain
LILA-Supermercado	Barquisimeto	Venezuela

6.3 SQL NULL Functions

Products

P_Id	ProductName	UnitPrice	UnitsInStock	UnitsOnOrder
1	Jarlsberg	10.45	16	15
2	Mascarpone	32.56	23	
3	Gorgonzola	15.67	9	20

```
SELECT ProductName, UnitPrice * (UnitsInStock + UnitsOnOrder)
FROM Products;
```

UnitsOnOrder **IS NULL** => UnitPrice * (UnitsInStock +
UnitsOnOrder) **IS NULL**

6.3 SQL NULL Functions

MySQL IFNULL()	SELECT ProductName, UnitPrice * (UnitsInStock + IFNULL(UnitsOnOrder, o)) FROM Products;
COALESCE()	SELECT ProductName, UnitPrice * (UnitsInStock + COALESCE(UnitsOnOrder, o)) FROM Products;
SQL Server ISNULL()	SELECT ProductName, UnitPrice * (UnitsInStock + ISNULL(UnitsOnOrder, o)) FROM Products;
MS Access IsNull()	SELECT ProductName, UnitPrice * (UnitsInStock + IIF(IsNull(UnitsOnOrder), o, UnitsOnOrder)) FROM Products;
Oracle NVL()	SELECT ProductName, UnitPrice * (UnitsInStock + NVL(UnitsOnOrder, o)) FROM Products;

6.3 SQL SELECT INTO

- Оператор SELECT INTO копіює дані з однієї таблиці в нову.
- The SELECT INTO statement copies data from one table into a new table.

```
SELECT column1, column2, column3, ...
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

6.3 SQL SELECT INTO

```
SELECT * INTO CustomersBackup2017
FROM Customers;
```

```
SELECT * INTO CustomersBackup2017 IN 'Backup.mdb'
FROM Customers;
```

```
SELECT CustomerName, ContactName INTO CustomersBackup2017
FROM Customers;
```

```
SELECT * INTO CustomersGermany FROM Customers
WHERE Country = 'Germany';
```

```
SELECT Customers.CustomerName, Orders.OrderID INTO CustomersOrder2017
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

6.3 SQL SELECT INTO

- SELECT INTO також можна використовувати для створення нової, порожньої таблиці, використовуючи схему іншої. Просто додайте вираз WHERE, в результаті якого запит не повертає даних.
- SELECT INTO can also be used to create a new, empty table using the schema of another. Just add a WHERE clause that causes the query to return no data.

```
SELECT * INTO newtable
FROM oldtable
WHERE 1 = 0;
```

6.3 SQL

INSERT INTO SELECT

Оператор INSERT INTO SELECT копіює дані з однієї таблиці та вставляє їх в іншу таблицю.

- INSERT INTO SELECT вимагає збігу типів даних у вихідних та цільових таблицях.
- Існуючі записи в цільовій таблиці не зазнають змін.

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

- INSERT INTO SELECT requires that data types in source and target tables match
- The existing records in the target table are unaffected

6.3 SQL

INSERT INTO SELECT

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ... FROM table1
WHERE condition;
```

Example 1:

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers;
```

Example 2:

```
INSERT INTO Customers (CustomerName, City, Country)
SELECT SupplierName, City, Country FROM Suppliers
WHERE Country='Germany';
```


Контрольні питання Assessment questions

1. SELECT
2. SELECT DISTINCT
3. WHERE
4. AND, OR, NOT
5. ORDER BY
6. INSERT INTO
7. NULL
8. UPDATE
9. DELETE
10. SELECT TOP
11. MIN, MAX
12. COUNT, AVG, SUM
13. LIKE
14. IN
15. BETWEEN
16. AS
17. JOIN
18. UNION
19. GROUP BY
20. HAVING
21. EXISTS
22. ANY, ALL
23. CASE
24. NULL Functions
25. SELECT INTO
26. INSERT INTO SELECT