Databaser

SQL & NoSQL

Vad är SQL?

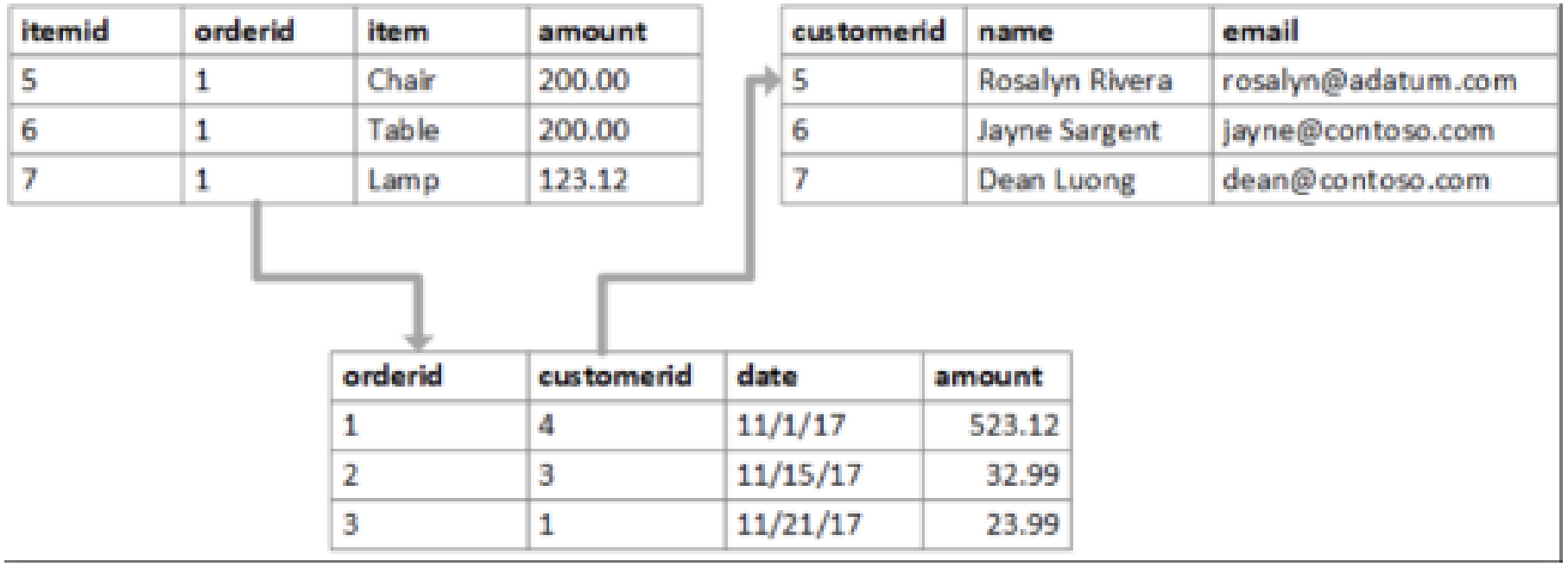
- SQL står för Structured Query Language.
- SQL är ett standardprogrammeringsspråk som är särskilt utformat för att lagra, hämta, hantera eller manipulera data i ett relationsdatabashanteringssystem (RDBMS). SQL blev en ISO-standard 1987.
- SQL utvecklades ursprungligen hos IBM i början av 1970-talet. Ursprungligen kallades det SEQUEL (Structured English Query Language) som senare ändrades till SQL (uttalas som S-Q-L).

Vad kan vi göra med SQL?

- Exempel:
- Du kan skapa en databas.
- Du kan skapa tabeller i en databas.
- Du kan fråga eller begära information från en databas.
- Du kan infoga poster i en databas.
- Du kan uppdatera eller ändra poster i en databas.
- Du kan radera poster från databasen.
- Du kan ställa in behörigheter eller åtkomstkontroll i databasen för datasäkerhet.

Relations-databas

- En relationsdatabas är en databas uppdelad i logiska enheter som kallas tabeller, där tabeller är relaterade till varandra i databasen. Relationsdatabasen gör att data kan delas upp i logiska, mindre och hanterbara enheter för enklare underhåll och bättre prestanda.



itemid	orderid	item	amount
5	1	Chair	200.00
6	1	Table	200.00
7	1	Lamp	123.12

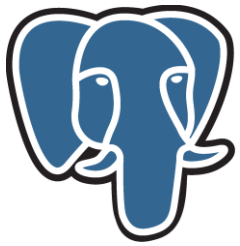
customerid	name	email
5	Rosalyn Rivera	rosalyn@adatum.com
6	Jayne Sargent	jayne@contoso.com
7	Dean Luong	dean@contoso.com

orderid	customerid	date	amount
1	4	11/1/17	523.12
2	3	11/15/17	32.99
3	1	11/21/17	23.99

Example of Relational Database Schema

Populära Relations-databaser (SQL)

- Microsoft SQL Server EXPRESS
- Oracle
- MySQL / MariaDB
- PostgreSQL
- Microsoft Azure SQL



Non-Relational Databases – (NoSQL)

- Till skillnad från sina relationsdatabaser (SQL), tillåter databaser med NoSQL mycket mer flexibilitet och anpassningsförmåga när du designar din applikation. Om dina krav inte är tydliga från början eller om du har att göra med stora mängder ostrukturerad data, kanske du inte har lyxen att utveckla en relationsdatabas med tydligt definierade tabeller och relationer.
- NoSQL-databaser är dokumentorienterade. Istället för att använda tabeller, tillåter dessa dokument oss att lagra ostrukturerad data i ett enda dokument.
- Så ett dokument kan innehålla en kunds information, samt alla deras beställningar hittills, deras favoriter etc. Detta är mer intuitivt och kräver färre ”hoppar” över tabellerna för att hitta all information som rör en kund. Observera dock att detta resulterar i ett behov av ytterligare bearbetningsarbete och mer lagring när dokumentstorlekarna växer.
- Lagringen kommer heller inte att vara så högt organiserad med en relationsdatabas.

Populära NoSQL databaser

- MongoDB
- Oracle NoSQL
- Apache CouchDB
- Redis

Azure Cosmos DB

redis

ORACLE

NOSQL DATABASE

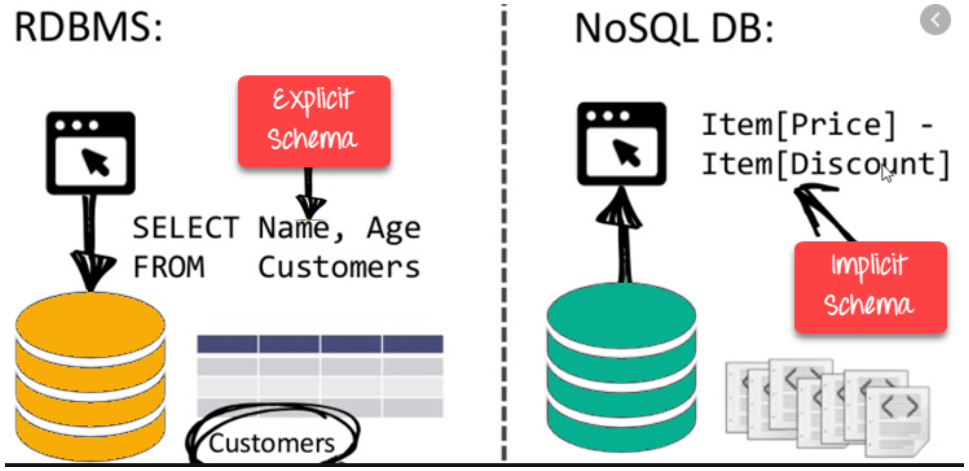
Vilken ska man välja?

- **Använd SQL när:**
- du kommer att tillämpa principerna för ACID (Atomicity, Consistency, Isolation, Durability). Detta minskar avvikelser, upprätthåller integritet och det är därför som detta föredras för handel och finansiella applikationer.

- din data struktur sällan ändras. Eller om den förväntas att inte förändras för framtida krav.
- **Atomicity** – Alla eller inga transaktioner görs
- **Consistency** – Transaktionen är klar eller tidigare tillstånd returneras
- **Isolation** – Transaktioner är isolerade från varandra
- **Durability** – Slutförd transaktion sparas säkert

Vilken ska man välja?

- **Använd NoSQL när:**
- du ska utveckla någonting snabbt.
- NoSQL stöder snabba förändringar I design och bra för kodnings sprintar när man jobbar Agilt.
- Och även när kravspec ändras ofta.
- Du lagrar stora mängder data med liten eller ingen struktur. Ungefär som uttryckt i föregående punkt, om dina datakrav inte är tydliga, men du vet att du måste lagra massor av data någonstans och på något sätt så kan du använda den här databastypen.



SQL syntax

- Exempel:

```sql
SELECT emp_name, hire_date, salary FROM employees WHERE salary > 5000;
```

- Detta är samma kod men mer läsbar:

```sql
SELECT emp_name, hire_date, salary
FROM employees
WHERE salary > 5000;
```

SQL kommentarer

- Exempel:

```sql
-- Select all the employees
SELECT * FROM employees;
```

- Multi-line kommentarer:

```sql
/* Select all the employees whose
salary is greater than 5000 */
SELECT * FROM employees
WHERE salary > 5000;
```

SQL – Skapa databas

- Exempel:

```
CREATE DATABASE database_name;
```

SQL – Skapa tables

- Nu när vi har lärt oss och skapa en databas så ska vi skapa några ”tables” i vår databas som kommer att innehålla vår data.

- En databas ”table” organiserar info i rader och kolumner.

```sql
-- Syntax for MySQL Database
CREATE TABLE persons (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15) NOT NULL UNIQUE
);

-- Syntax for SQL Server Database
CREATE TABLE persons (
    id INT NOT NULL PRIMARY KEY IDENTITY(1,1),
    name VARCHAR(50) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15) NOT NULL UNIQUE
);
```

Datatyper

- https://www.w3schools.com/sql/sql_datatypes.asp

Vad är en begränsning (constraint)?

- En begränsning är helt enkelt en begränsning som placeras på en eller flera kolumner i en tabell för att begränsa typen av värden som kan lagras i den kolumnen. Begränsningar tillhandahåller en standardmekanism för att bibehålla noggrannheten och integriteten för data i en databastabell.

- Det finns flera olika typer av begränsningar i SQL, inklusive:
- NOT NULL
- PRIMARY KEY
- UNIQUE
- DEFAULT
- FOREIGN KEY

NOT NULL Constraint

- Begränsningen NOT NULL anger att kolumnen inte accepterar NULL-värden
- Detta innebär att om NOT NULL-begränsning tillämpas på en kolumn kan du inte infoga en ny rad i tabellen utan att lägga till ett NOT NULL-värde för den kolumnen
- Detta innebär att de tre kolumner, id, namn och telefon inte accepterar NULL-värden:

```
1  CREATE TABLE persons (
2      id INT NOT NULL,
3      name VARCHAR(30) NOT NULL,
4      birth_date DATE,
5      phone VARCHAR(15) NOT NULL
6  );
```

PRIMARY KEY(PK) Constraint

- PRIMÄR KEY-begränsningen identifierar att kolumnen eller uppsättningen kolumner som har värden som unikt identifierar en rad i en tabell.
- Du kan inte ha två rader i en tabell som har samma PRIMARY KEY.
- Du kan inte heller ange ett NULL-värde i en PRIMARY KEY.

```sql
CREATE TABLE persons (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15) NOT NULL
);
```

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

UNIQUE Constraint

- UNIQUE-constraint begränsar en eller flera kolumner så att de innehåller unika värden i en tabell.
- använd en UNIQUE-begränsning istället för en PRIMARY KEY-begränsning när du vill genomdriva det unika med en kolumn, eller en kombination av kolumner, som inte är den PRIMARY KEY.

```sql
CREATE TABLE persons (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15) NOT NULL UNIQUE
);
```

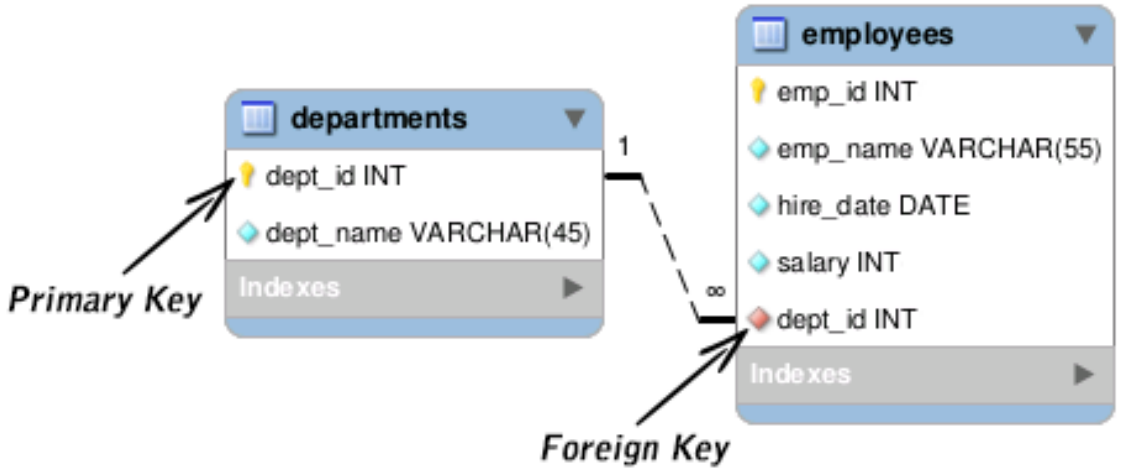
DEFAULT Constraint

- Andvänd Default när du vill sätta ett default värde när insättning sker t.ex:

```sql
CREATE TABLE persons (
    id INT NOT NULL PRIMARY KEY,
    name VARCHAR(30) NOT NULL,
    birth_date DATE,
    phone VARCHAR(15) NOT NULL UNIQUE,
    country VARCHAR(30) NOT NULL DEFAULT 'Australia'
);
```

FOREIGN KEY (FK) Constraint

- En foreign key (FK) är en kolumn eller kombination av kolumner som används för att upprätta en relation mellan data i två tabeller:



Visa data från table

```
SELECT * FROM table_name;
```

Output:

```
+--------+--------------+------------+--------+---------+
| emp_id | emp_name     | hire_date  | salary | dept_id |
+--------+--------------+------------+--------+---------+
|      1 | Ethan Hunt   | 2001-05-01 |   5000 |       4 |
|      2 | Tony Montana | 2002-07-15 |   6500 |       1 |
|      3 | Sarah Connor | 2005-10-18 |   8000 |       5 |
|      4 | Rick Deckard | 2007-01-03 |   7200 |       3 |
|      5 | Martin Blank | 2008-06-24 |   5600 |    NULL |
+--------+--------------+------------+--------+---------+
```

SQL functions

- https://www.w3schools.com/sql/sql_ref_sqlserver.asp

CREATE TRIGGER

```
CREATE TRIGGER [schema_name.]trigger_name
ON table_name
AFTER  {[INSERT],[UPDATE],[DELETE]}
AS
{sql_statements}
```

Labbar

- 1. Installera SQL server (youtube video): Länk
- 2. Gör denna labb på microsoft (SQL): Länk
- 3. Skapa en databas(SQL kod finns på sida 12 & 13) **lokalt**.
- 4. Sedan gör du precis som **punkt 3,** men denna gång ska databasen vara i Azure/molnet.
- 5. När du är klar med **punkt 4** så gör du denna lab (NoSQL): Länk
- 6. Om du klar med detta så kan du börja titta på monogDB: Länk (Valfritt)
- *OBS! Glöm inte att skapa en ”virtual environment”. Beskrivning om hur vi gör detta finns inspelad från dagens lektion samt att det kommer dyka upp ett **Dokument om instruktioner** innan dagens slut. **01/25-2021***

Tveka inte att höra av er om ni har funderingar.