Databases

בסיס נתונים. מסד נתונים.

"מהמסד ועד הטפחות"

מסד – יסודות

טפחות: פינות של הצריחים בגג

בסיס הנתונים, הוא בעצם הבסיס לכל מערכת מידע. אין שום משמעות לקוד שנכתוב, אם לא יהיו נתונים בבסיסו.

נתונים נשמרים בדיסק כלשהו. פעם זה היה בסרטים מגנטיים. לפני זה זה היה בכרטיסיות מנוקבות.

הגענו לשלב באבולוציה של פיתוח מערכות מידע, אזור שנות ה-70 המוקדמות, לא היו דטהביסיים, כל מערכת מידע הייתה מנהלת את המידע בקבצים.

גיבוי, שחזור, נפח (מה קורה כשנגמר המקום בדיסק וכו')

פותחו תוכנות ספציפיות לניהול מידע (על גבי קבצים) שנקראו DBMS: Database Management System או בקיצור: database

היו שני דורות מוקדמים של דטהבייסים שלא מעניינים אותנו. לקראת סוף שנות ה-70' התקבעה מתודולוגיית ה-relational database או בעברית דטהבייס רלציוני. ובראשי תיבות : RDBMS.

זו השיטה שהתקבעה מאז סוף שנות ה-70 ועד +- 2010

בשנות ה-70 המשאב היקר ביותר היה שטח איחסון ולכן גישת ה-RDBMS הייתה חסכון בשטח אחסון.

כדי לחסוך בשטח אחסון, המציאו מושג שנקרא נורמליזציה של נתונים, שהשפיע על הדרך שבה עבדו דטהבייסים עד 2010 פחות או יותר (ו-RDBMS-ים עובדים כך עד היום)

פחות או יותר בשנת 2010 התחילו לצוץ כל מיני דטהבייסים מדור 4 (להזכירכם RDBMSזה הדור ה-3 של דטהבייסים).

מה קרה ב-2010?

האינטרנט התחיל להתפוצץ, והתגלה שלא ניתן להגיע ל-scale גבוה עם גלי RDBMS והתחיל הדור ה-4 של ה-database-ים שנקרא בפשטות: NoSQL

SQL היא שפת התכנות של ה-RDBMS והכוונה ב-NoSQL היא שאנחנו בונים DBMS-ים שלא תומכים במתודולוגיות ה-RDBMS ה"מיושנות".

עדיין נעשה שימוש נרחב מאד ב-RDBMS כי לא כל מערכת שנפתח, צפויה לגדול לרמה של פייסבוק. אם למשל אני מפתח מערכת לעיריית ראש העין, אפשר בקלות לפתח אותה על RDBMS.

אם אני מפתח את מרשם האוכלוסין של הודו, אולי לא אבחר ב-RDBMS

ההבדל העיקרי בין RDBMS ל-NoSQL:

RDBMS דוגל במתודולוגיה המכונה: ACID

Atomicy, Consistency, Isolation, Durability

במערכות NoSQL, נהוג לוותר על אחד מה-4 ולתמוך רק ב-3.

מערכות NoSQL יהיו בזבזניות בשטח אחסון, מהירות מאד, אך יפסידו את אחד מערכי ה-ACID.

שמות RDBMS שאולי אנחנו מכירים : MySQL, SQL Server, SQLite, Postgres, IBM DB2, Oracle

שמות NoSQL: MongoDB, , Firebase, , DynamoDB, CouchDB, Cassandra, Elastic

נתקין XAMPP

LAMP

Linux, Apache, Mysql, PHP/Perl/Python

הסטאק הטכנולוגי שבאמצעותו פיתחו אתרי אינטרנט בתחילת עידן האינטרנט.

היו בעצם 3 סטאקים עיקריים:

* מייקרוסופט ASP (Active server pages) על שרת שנקרא IIS (Internet Information Server) – עלה כסף. הרבה כסף.
* J2EE – Java 2 Enterprise Edition – היה חינם, אך דרש לפתח בשפת Java שהיא שפה די קשה ומורכבת, והצריך שרתי unix שהם בעצמם היו שרתים יקרים (למשל שרתי Solaris של חברת Sun Microsystem)
* LAMP – היה חינם. וכל מה שדרש שזה שרת לינוקס קטן. ולכן זו הייתה הסביבה/סטאק טכנולוגי הפופולרי ביותר לפיתוח אתרי אינטרנט. פייסבוק למשל פותחה ב-lamp, ebay פותחה ב-j2ee, אמאזון התחילה lamp

מה זה xampp? זהו סטאק של lamp למחשבים שאינם linux. קרי ווינדוס ומק.

X – cross platform

Apache – web server

Mysql – RDBMS

Php/perl/python – scripting language

ממה מורכב דטהבייס רלציוני

טבלה – זוהי היישות העיקרית בדטהבייס. אפשר להתייחס אליה לוגית כקובץ שמכיל רשומות.

בתוך טבלה יהיו שורות/רשומות שמכילות את המידע עבור ישות ספציפית.

כל שורה מחולקת לעמודות.

מבנה השורה הוא קבוע פר טבלה. כלומר, אנו מגדירים מראש איך ייראה מבנה השורה כשאנו יוצרים את הטבלה. למבנה הטבלה נהוג לקרוא גם schema

בכל עמודה (תא) – יהיה רק ערך אחד בודד. לא שומרים מערכים או ערכים מרובים בתוך תא בודד.

Primary key – לכל שורה בטבלה, צריך שיהיה primary key שהוא ערך ייחודי שבאמצעותו ניתן לזהות את השורה בצורה חד חד ערכית. אחד מהתאים בשורה ישמש ב-primary key. נהוג לקרוא לו id.

Primary key מיושם כ-unique index

Foreign key – עמודה שהערכים שלה, צריכים להיות מתאימים ל-Primary key כלשהו בטבלה אחרת

גם primary key וגם foreign key הם אינדקסים. אינדקס הוא מושג טכני של דטהבייס, בפועל הדטהבייס שומר לעצמו סוג של העתק של הנתונים של עמודת האינדקס לצד הטבלה, ואת החיפושים, אם צריך, הוא עושה על האינדקס במקום על הטבלה. אינדקס נועד לאפשר לדטהבייס למצוא נתונים בצורה יעילה.

Id – identification

אנחנו יכולים לבחור את טיפוס הנתונים שבו נשתמש למפתח הראשי, באפליקציות אמיתיות בעידן המודרני נהוג להשתמש במחרוזת רנדומלית ארוכה כלשהי. למשל : 2eadd915-46d7-4c19-874f-576208551e69

ל-mysql יש שיטה מאד קלה של מספור אוטומטי רץ הנקראת autoincrement, היא מייצרת ids מספריים 1,2,3 וכן הלאה. נשתמש בה לצורך הלימוד. לא נשתמש בה במציאות.

קטגוריות של טבלאות:

1. טבלת נתונים – core table, base table טבלה ששומרת את נתוני האפליקציה שלנו, טבלה זו תלך ותגדל ככל שהאפליקציה תלך ותגדל.
2. טבלת חיפוש – lookup table מקביל ל-dropdown בממשקי משתמש. אלו טבלאות שבהן יהיה סט סגור של נתונים, למשל שמות ערים, למשל סוגי טיפוסים, למשל מין,
3. טבלת child או טבלת details והיא מתארת ריבוי ערכים אפשרי עבור ישות בודדת מטבלת core. למשל ציונים של תלמידים, טלפונים של תלמידים. ספרים של סופרים, שירים של זמרים. בניגוד ל-lookup table לא מדובר בסט סגור וסופי של נתונים אלא בלתי מוגבל
4. טבלת association. טבלת קישור עבור יחסי many to many, טבלה זו ייחודית בכך, שאין בה עמודת primary key אלא ה-primary key מורכב משתי עמודות שונות.

Relations

Relations הם יחסים בין טבלאות שונות שנועדו לשמור על data integrity קרי שלמות הנתונים. צריך לשמור על דטה אינטגריטי ב-3 מצבים:

* הוספת רשומה. עלינו לוודא שאיננו מכניסים cityId שאינו קיים בטבלת cities
* עדכון רשומה. עלינו לוודא שאיננו מעדכנים cityId שאינו קיים בטבלת cities
* מחיקה. עלינו לוודא שאיננו מוחקים city שמקושר לתלמיד שלו יש cityId כזה.

ישנם גם 3 סוגי realtions:

1. One to many – רשומת primary key אחת מקושרת למספר רשומות foreign key. למשל רשומת סטודנט אחת מקושרת למספר רשומות טלפונים. למשל רשומת עיר אחת מקושרת למספר רשומות סטודנטים. זה בעצם היישום גם ב-lookup וגם ב-child.
2. One to one – מדי פעם נרצה לקשר רשומת primary key אחת לרשומת foreign key אחת. (מדובר במצב די נדיר, אפר לחשוב עליו כמקרה קצה של one to many כאשר בצד של ה-many יש רק רשומה אחת.) למשל סטודנט וכרטיס אשראי. למשל (לפחות בישראל) בעל ואשה.
3. Many to many – זהו קשר המבטא ריבוי מפתחות זרים כבור מפתחות ראשיים, וגם הפוך. למשל: סטודנטים וקורסים. סטודנט יכול להיות רשום למספר קורסים. ולכן לסטודנט אחד, יש קשר להרבה קורסים. מצד שני, בכל קורס, לומדים מספר סטודנטים, ולכן לכל קורס יש קשר למספר סטודנטים.