

## מטרת הקורס

1. עיצוב מסדי נתונים DB Design  
איך מסדי נתונים מאורגנים.
2. שפות מסדי נתונים DB Languages  
איך לגשת לבסיסי הנתונים
3. מימוש DBMS implementation  
מימוש מערכות שונות של בסיסי נתונים

## הגדרה

אין טעם לתת הגדרה מדוייקת לבסיס נתונים(למרות שיש הגדרה כזו), אבל בסיס נתונים כולל:

- בסיס נתונים DB
  - אוסף של נתונים
  - קשרים בין הנתונים
- תוכנה DBMS (DB Management System)
  - שניהם ביחד נקראים DBS (DB System).
  - בקורס הזה נלמד על שניהם - גם איך לארגן נתונים בבסיס, וגם איך לטפל בהם.

## תכונות של DBMS

1. מודל נתונים Data model  
כל DBMS מבוסס על איזשהו מודל(היררכי, יחסי וכו')
2. שפה - HLL-High Level Language
3. ניהול טרנסקציות Transactions management
4. אבטחת נתונים Data security
5. אפשרות לשחזר נתונים - Durability(restoration)
6. הפרדה - Data independence

## קצת היסטוריה

התחום התחיל ב1960. אמנם גם לפני כן עבדו עם נתונים - אבל בתור מערכות קבצים (File systems). מערכות קבצים לא תומכות ברוב התכונות - הן מאפשרות רק לאחסן קבצים. לכן בשנות ה60 בנו DBMS ראשונות, שהיו מבוססות על מודל נתונים בצורת עץ - בסיס נתונים היררכי.

לבסיס נתונים בצורת עץ יש כמה חסרונות, בין השאר:

- אין אלגוריתם פורמלי שמאפשר לבנות בסיס כזה. מי שבונה את הבסיס עושה את זה לפי ההבנה והנסיון שלו, אבל אין אפשרות לבדוק שהוא עשה את זה בצורה נכונה, יעילה. אין אפשרות לבנות בסיס נתונים כזה בצורה סטנדרטית, פורמלית.
  - שפת השאילתות במודל כזה היא שפה פרוצדורלית - Procedural Query Language. חייבים לתאר איך להגיע לתוצאה.
  - אי אפשר לתאר קשרי רבים לרבים - רק אחד לרבים ( $1 : N$ ). לכל קודקוד יכולים להיות הרבה בנים, אבל לכל בן רק אב אחד.
- את החסרון הזה ניתן לבנות באמצעות בניית מודל רשתי Network, שמבוסס על גרף. כאן ניתן לתאר קשרים מסוג רבים לרבים, אבל שני החסרונות הקודמים (והכי חשובים) עדיין נשארים.

### המהפכה

בשנת 1970 התחוללה מהפכה בעולם של מסדי הנתונים. פורסם מאמר מדעי - "A Relational Model for Large Shared Data Banks/Codd". זו לא הייתה אימפלמנטציה - אלא גישה חדשה, שאפשר לשמור נתונים לפי מודל אחר, המבוסס על תורת הקבוצות, ואפשר לראות את המודל הזה באמצעות טבלאות. המודל הזה נקרא "מודל יחסי (Relational)" או "מודל טבלאי".

במודל הזה, חוץ מזה שאפשר לתאר קשרים מסוג רבים לרבים, אפשר לתאר אלגוריתם מדויק לבניית סכימה - algorithm for schema. בנוסף, שפת השאילתות הייתה דקלרטיבית - declarative QL, שבה לא צריך לכתוב פרוצדורה, אלא לכתוב הצהרה, מה רוצים לקבל.

### 1990

המודל הרילציוני נשאר, אבל משלבים אותו עם OOP, עם אינטרנט (למשל XML) וכו'. אבל המודל הרילציוני נשאר הבסיס.

## Data Model

## רמות הפשטה - Levels of abstraction

בכל DBMS יש רמות שונות של הפשטה.

1. רמה הפיסית - ברמה הפיסית מתואר איך הנתונים מאוכסנים בדיסק(קבצים). זה תיאור מאוד מדויק, עם כל הטיפוסים, גודל, שיטת גישה, אינדקס וכו' - כל הפרטים שלא מעניינים למשתמש.
  2. רמה קונספטואלית(conceptual) - מתוארים אותם נתונים, אבל בלי כל הפרטים האלה. מתואר הארגון הלוגי של הנתונים, מה הקשרים ביניהם. לא כל הנתונים מעניינים את כולם, ולא לכולם מותר לגשת לכל הנתונים.
  3. רמת תצפית(view) - תצפית על הנתונים. משתמש שיש לו view מסוים, יכול לראות חלק מהנתונים - החלק שרלוונטי אליו ושמותר לו לגשת אליו. למשתמשים שונים בונים view שונים.
- אלו רמות הפשטה של כל מודל - כל מודל(היררכי, רילציונלי וכו') ניתן לראות ברמות ההפשטה האלה.

## HLL - High Level Language

בDB נתונים מאוכסנים לטווח ארוך. לכן יש הבדל בין בניית המבנה של המאגר לבין העבודה עם הנתונים עצמם. לכן בעולם של מסדי נתונים מבדילים בין שני סוגים של שפות:

1. שפה להגדרת נתונים - DDL - Data Definition Language
2. שפה לפעולות על הנתונים - DML - Data Manipulation Language

### DDL

בDDL עושים שני דברים:

1. תיאור של סכימה - schema description

```
CREATE TABLE enroll
(ID: INTEGER,
Name: CHAR(10),
Course: CHAR(6),
Sem: CHAR(1),
Grade: INTEGER);
```

2. עדכון של סכימה - schema update

בDDL לא מתוארים הנתונים, אלא מבנה הטבלה - שם הטבלה, שמות של שדות, תכונות של שדות וכו'. לא מתואר תיאור פיזי - לא כתוב איך לשמור את הנתונים, באיזה קובץ וכו' - על זה הDBMS אחראית. עם הDDL עובדים לעיתים מאוד רחוקות - רק כאשר יש צורך בעדכון של מבנה הנתונים.

### DML

עבודה עם נתונים היא עבודה מאוד דינאמית, יומיומית, ולה יש סוג אחר של שפות - DML.

## SQL - Structural QL

באותה שפה, SQL, יש חלק שהוא DDL וחלק שהוא DML.

### מי משתמש בשפות האלה?

בDDL המשתמש הרגיל לא נוגע - יש אדם מיוחד, DB admin, שבונה את הסכימות, את הviewים.

המשתמש הרגיל כותב תוכניות בDML.

### דוגמאות

#### דוגמה 1 - שליפת נתונים

```
SELECT course
FROM enroll
WHERE ID=123;
```

השאלתה מגדירה מה לקחת(course), מאיפה לקחת(טבלת enroll), ולפי איזה תנאי(id=123)

#### דוגמה 2 - שינוי נתונים

```
UPDATE enroll
SET grade=grade+5
WHERE course='DB' AND ID=123;
```

השאלתה מגדירה על איזה טבלה לפעול(enroll), מה לעשות שם(להעלות ציון ב5), ומה התנאים(course='DB' AND ID=123)

### שפה מארחת - Host Language

יש דברים שעושים מחוץ לDB - למשל Interface - ואותם לא עושים בDDL או בDML, אלא בשפה מארחת, שיכולה להיות c, java, basic או כל שפה אחרת. השפה המארחת שולחת את פקודות הSQL לDBMS.

## Transactions management

כיוון שהרבה משתמשים פונים לאותם נתונים בו זמנית, צריך לנהל את העבודה עם הנתונים. צריך לתמוך בשני דברים:

1. הפרדה isolation

צריך לגרום לכך שכל טרנזקציה תרגיש כאילו היא עובדת לבד - למרות שעוד טרנזקציות עלולות לגשת לאותם נתונים.

2. אטומיות atomicity

אם באיזשהו מקום צריך להפסיק את הטראנזקציה, אסור שתתבצע חצי עבודה. או שהטראנזקציה סיימה לעבוד - או שהיא לא עשתה כלום.

זהו תפקיד ה-DBMS לספק מצב קונסיסטנטי בכל רגע ורגע, כדי שכל התוצאות יהיו נכונות.

## Data security

אין צורך להסביר למה זה חשוב.

## Durability

בסיסי נתונים הם מאגרים מאוד גדולים, ואם נופל משהו צריכה להיות אפשרות לשחזר את הנתונים לאיזשהו מצב קונסיסטנטי(לא בהכרח אחרון) שמשם אפשר להתחיל. יש כלים שמאפשרים את זה - למשל גיבוי (copy). אבל אי אפשר לעשות גיבוי כל רגע, ולכן בנוסף לזה רושמים את כל הטראנזקציות ב-log(יומן). ככה אפשר לשחזר נתונים ממצב קונסיסטנטי כלשהו באמצעות ה-log.

## Data independence

האפליקציה לא תלויה בשיטת האחסון ובשיטת הגישה לנתונים.