MASARYKOVA UNIVERZITA
INSTITUT BIOSTATISTIKY A ANALÝZ

Databázové systémy a SQL

Lekce 5

Daniel Klimeš

Databázové objekty

- Nejvýznamnější databázové objekty
  - Tabulky (tables)
  - Pohledy (views)
  - Indexy (indexes)
  - Sekvence (sequences)
  - Procedury (procedures)
  - Funkce (functions)
  - Triggery (triggers)

Informace o objektech jsou uloženy v metadatech (systémových datech) databáze
Přístup k nim je databázově specifický

ORACLE
- systémové tabulky – uživatelům pouze pro čtení
- metatabulka o metatabulkách – DICTIONARY
- tabulky USER_XXX – objekty vytvořené uživatelem
- tabulky ALL_XXX – objekty přístupné uživateli
- tabulky DBA_XXX – všechny objekty databáze – přístupné jen administrátorovi

Hierarchie objektů

•Schéma = sada databázových objektů patřící obvykle jednomu projektu/ podprojektu

•ORACLE
- Na serveru je 1 databáze
- Každý uživatel má automaticky své schéma
- Uživatel vytváří objekty ve svém schématu

•PostgreSQL
- Na serveru je N databází
- V každé databázi je jedno výchozí schéma *public*
- V každé databázi je možné vytvářet další schémata
- Uživatel vytváří objekty v libovolném schématu, defaultně v public

- Odkaz na objekt ve schématu: schema.objekt
  např.: student.patients

Tabulky

- Metatabulky USER_TABLES, ALL_TABLES, DBA_TABLES
  - sloupec table_name
- metatabulka TAB
  - sloupec tname, tabtype
- Sloupce tabulky - USER_TAB_COLUMNS
  - table_name, column_name, data_type
- PostgreSQL (ANSI standard)
  - information_schema.tables
  - information_schema.columns

DDL příkazy pro manipulaci s tabulkami

- CREATE TABLE
- DROP TABLE
- ALTER TABLE
- RENAME TABLE

Pohledy (VIEWS)

- Pohled = uložený SQL dotaz
- Pracuje se s ním stejně jako s tabulkou
- Ve většině případů je možný pouze SELECT

- CREATE VIEW v_ukazka AS
  SELECT ps.patient_id, study_name FROM patient_study ps, studies s
  WHERE ps.study_id = s.stud y_id

```
SELECT study_name, count(*) FROM v_ukazka
GROUP BY study_name
```

DDL pro pohledy:

CREATE OR REPLACE VIEW AS
DROP VIEW

ORACLE metadata
- user_views, tab

PostgreSQL/ANSI
- information_schema.views

Indexes

•Indexy jsou obdobou kartotéky
• Umožňují rychlejší vyhledávání záznamů ve velkých tabulkách
• Urychlují SELECT dotazy, zpomalují INSERT, UPDATE, DELETE

• Indexy se vytváří nad jedním nebo více sloupci tabulky
• Standardně nad primárním klíčem a cizími klíči
• Dále nad sloupci, které se často používají za WHERE

•DDL pro indexy
  • CREATE INDEX
  • DROP INDEX
  • ALTER INDEX

ORACLE metadata
  user_indexes
  ind

Sekvence

•Sekvence generují za všech okolností unikátní čísla – posloupnost
• Použití pro primární klíče při insertech nových řádků

• SELECT jmeno_sekv.NEXTVAL from DUAL
• SELECT jmeno_sekv.CURRVAL from DUAL

• Každé zavolání NEXTVAL vrátí další číslo v posloupnosti bez ohledu na transakce
Při neúspěšném použití vygenerovaného ID vznikají “díry” v posloupnosti

•ORACLE DDL
  • CREATE SEQUENCE
  • DROP SEQUENCE
  • ALTER SEQUENCE
• ORACLE metadata
  • user_sequences

•PostgreSQL/ANSI
  • information_schema.sequences

TRIALDB - uložení dat

Uložení vlastních dat v TrialDB není klasický relační datový model
Generalizovaný model – EAV model – Entity – Attribute - Value

•Klasický datový model

ID pacienta	Hmotnost	Výška
PacientXY	84	186

EAV model

Entita	Atribut	Hodnota
Pacient	Otázka	Hodnota
PacientXY	Hmotnost	84
PacientXY	Výška	186

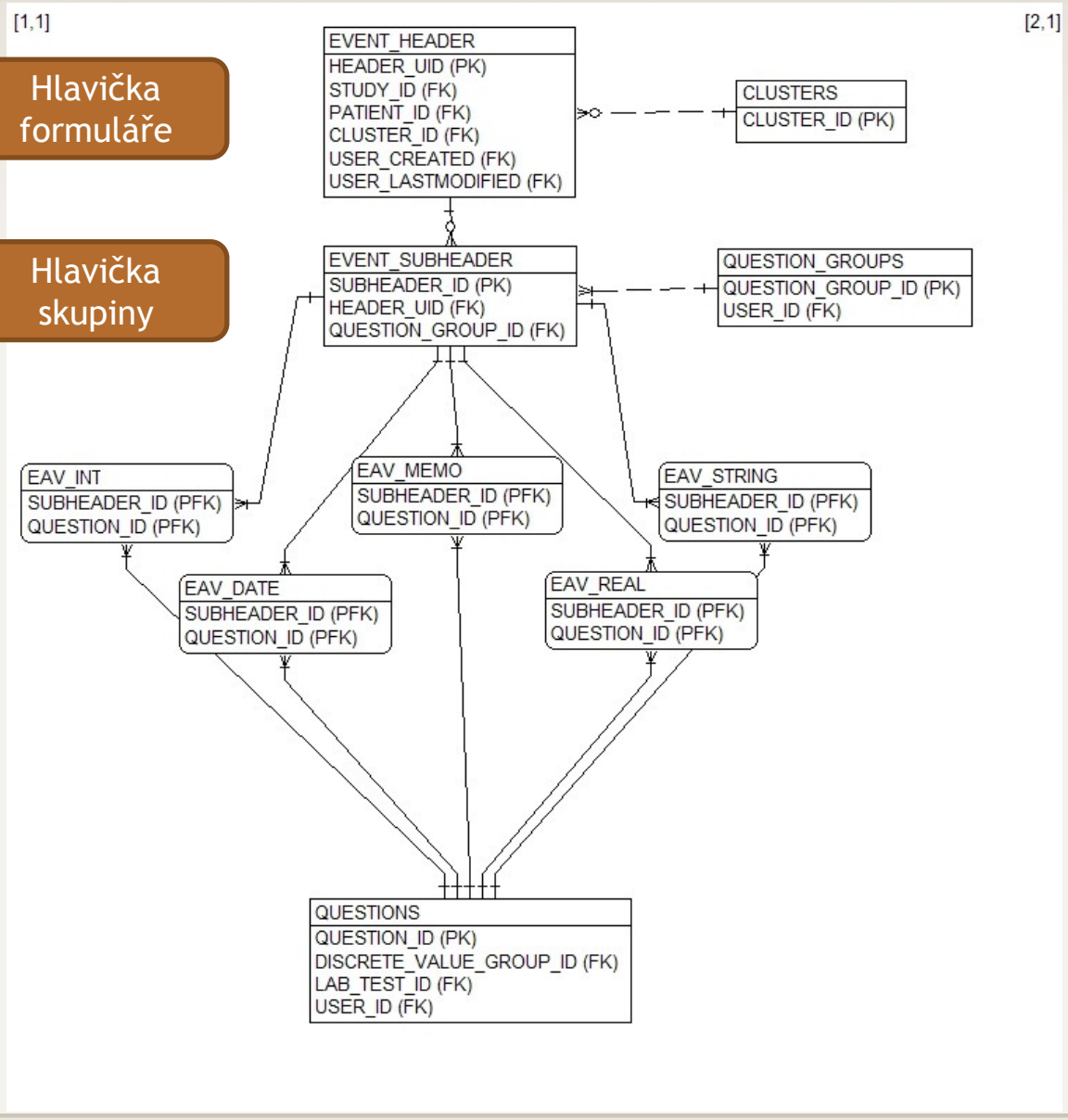
V datovém modelu TrialDB – rozpracované pro jednotlivé datové typy

TRIALDB - datový model

Hlavička formuláře

Hlavička skupiny

Vlastní data



Tabulka EVENT_HEADER

•Tabulka EVENT_HEADER – 1 řádek = 1 vyplněný formulář

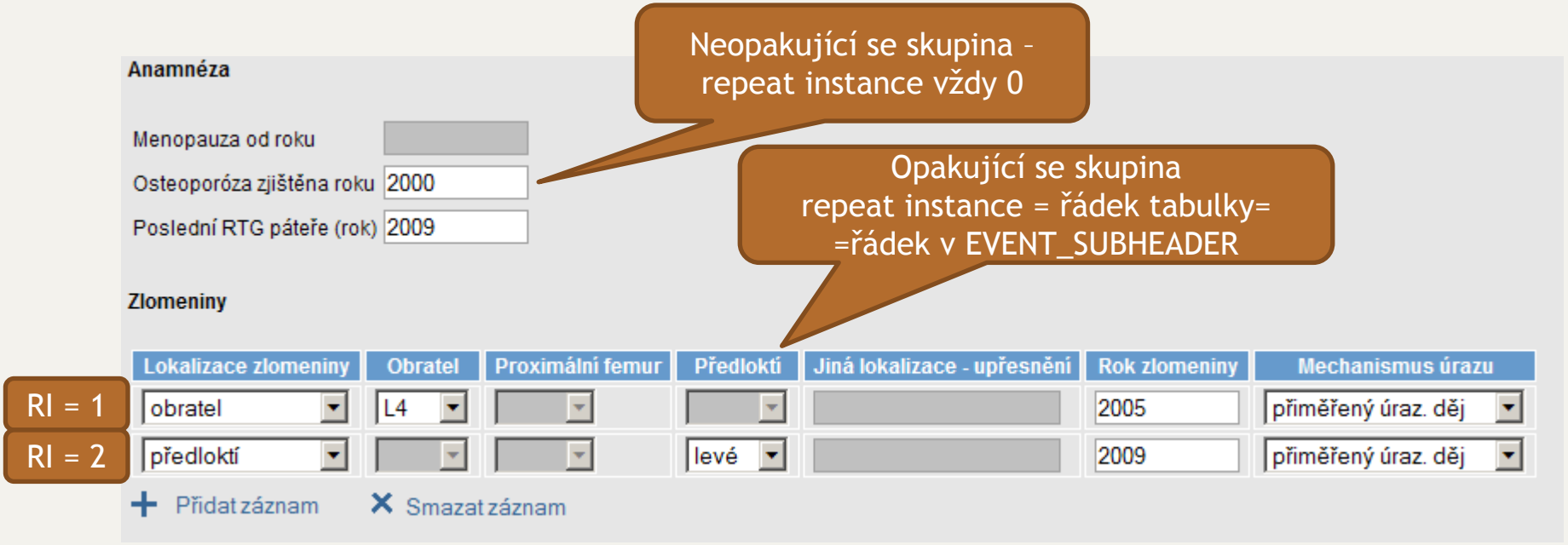
EVENT_HEADER
HEADER_UID (PK)
STUDY_ID (FK)
PATIENT_ID (FK)
PHASE_ID
CLUSTER_ID (FK)
DATE_COLLECTED
VERIFIED
COMPLETED
PROBLEMS
DATETIME_LAST_MODIFIED
USER_CREATED (FK)
USER_LASTMODIFIED (FK)
LOCKED
LOCKED_DATETIME
FORM_IDENT
FORM_IDENT2
FORM_IDENT3
FORM_IDENT4
IMPORT_ID
PAID_DATETIME
PARENT_HEADER_UID
CHILD_HEADER_TEXT

•HEADER_UID – primární klíč, generovaný
• STUDY_ID – klíč ke studii/registru
• PATIENT_ID – klíč k pacientovi
• PHASE_ID – klíč k fázi
• CLUSTER_ID – klíč k popisu formuláře
• DATE_COLLECTED – datum vyplnění formuláře
• DATETIME_LAST_MODIFIED
datum poslední změny dat ve formuláři

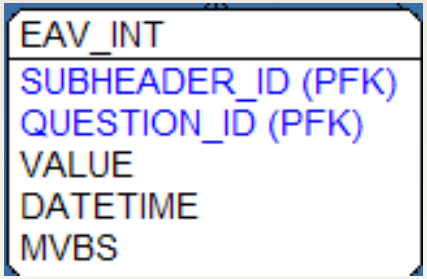
Tabulka EVENT_SUBHEADER

EVENT_SUBHEADER
SUBHEADER_ID (PK)
HEADER_UID (FK)
QUESTION_GROUP_ID (FK)
REPEAT_INSTANCE
EVENT_TYPE
START_OF_EVENT
END_OF_EVENT
INSTANCE_LABEL
LAB_ID
DATETIME_LAST_MODIFIED
USER_LASTMODIFIED
START_OF_EVENT_MVBS
END_OF_EVENT_MVBS
IMPORT_ID

- EVENT_SUBHEADER – 1 řádek = 1 vyplněná skupina otázek
  - SUBHEADER_ID – primární klíč, generovaný
  - HEADER_UID – klíč k vyplněnému formuláři (EVENT_HEADER)
  - QUESTION_GROUP_ID – klíč k popisu skupiny otázek
  - REPEAT_INSTANCE
    pořadové číslo vyplněné skupiny na formuláři



Tabulky EAV_XXX



- EAV_XXX – 1 řádek = 1 vložená hodnota
- SUBHEADER_ID + QUESTION_ID
  složený primární klíč
- SUBHEADER_ID klíč ke skupině (EVENT_SUBHEADER)
- QUESTION_ID – klíč k definici otázky
- DATETIME – datum a čas vyplnění
- MVBS – údaje o přesnosti či chybějící hodnotě
- VALUE – vlastní vyplněná hodnota

Cvičení

- Najděte formulář bez definované skupiny otázek
- Najděte formulář s největším počtem skupin
- Najděte formulář s největším počtem otázek
- Najděte nejčastěji používaný číselník

Cvičení

- WHERE **EXISTS** (SELECT * FROM….
- WHERE **NOT EXISTS** (SELECT * FROM…

- Vypište všechny sloupce tabulky student pro ty studenty, kteří mají zapsaný předmět Bi5447, ale nemají zapsán předmět Bi3030

```
SELECT * FROM student
   WHERE EXISTS
      (SELECT * FROM vyuka
        WHERE vyuka.predmet_id = 'Bi5447' AND
        student.student_uco = vyuka.student_uco
      )
   AND NOT EXISTS
      (SELECT * FROM vyuka
        WHERE vyuka.predmet_id = 'Bi3030' AND
        student.student_uco = vyuka.student_uco
      )
```

Cvičení

- Kolik vyplněných desetinných čísel obsahuje registr study_id = 3
  - tabulky EAV_REAL, EVENT_HEADER, EVENT_SUBHEADER

- Spojte uvedené tabulky dle klíčů – vnitřní spojení

```
SELECT * FROM event_header eh, event_subheader es, eav_real er
WHERE eh.header_uid = es.header_uid AND es.subheader_id = er.subheader_id
```

- Přidejte podmínku na konkrétní studii a spočítejte řádky

```
SELECT COUNT(*) FROM event_header eh, event_subheader es, eav_real er
WHERE eh.header_uid = es.header_uid AND es.subheader_id = er.subheader_id
AND eh.study_id = 3
```

Cvičení

- Kolik těchto hodnot je záporných?

```
SELECT COUNT(*) FROM event_header eh, event_subheader es, eav_real er
WHERE eh.header_uid = es.header_uid AND es.subheader_id = er.subheader_id
AND eh.study_id = 3 AND er.value < 0
```

- Kolik je to unikátních otázek (question_id)

```
SELECT COUNT(distinct question_id)
FROM event_header eh, event_subheader es, eav_real er
WHERE eh.header_uid = es.header_uid AND es.subheader_id = er.subheader_id
AND eh.study_id = 3 AND er.value < 0
```

Cvičení

- Jaké jsou průměrné hodnoty a směrodatná hodnota jednotlivých otázek? (vynechte záporné hodnoty) – QUESTION_ID + agregační funkce

```
SELECT question_id, AVG(value), STDDEV(value), MIN(value), MAX(value)
FROM event_header eh, event_subheader es, eav_real er
WHERE eh.header_uid = es.header_uid AND es.subheader_id = er.subheader_id
AND eh.study_id = 3 and er.value > 0
GROUP BY question_id
```

- Doplňte k seznamu název otázky

```
SELECT q.question_id, q.question_description, AVG(value), STDDEV(value)
 FROM event_header eh, event_subheader es, eav_real er, questions q
WHERE eh.header_uid = es.header_uid AND es.subheader_id = er.subheader_id
AND eh.study_id = 3 and er.value > 0 AND er.question_id = q.question_id
GROUP BY q.question_id, q.question_description
```

Cvičení

- Vypište hodnoty otázky question_id = 161 (PATIENT_ID, VALUE) pro všechny založené formuláře cluster_id = 65, study_id = 3

- Kolik je formulářů cluster_id = 65 , study_id = 3?

```
SELECT COUNT(*) FROM event_header WHERE study_id = 3 AND cluster_id = 65
```

```
1) SELECT eh.patient_id, er.value
FROM event_header eh LEFT JOIN
     (event_subheader es INNER JOIN eav_real er
      ON es.subheader_id = er.subheader_id AND er.question_id = 161)
    ON eh.header_uid = es.header_uid
WHERE  eh.study_id = 3 AND eh.cluster_id = 65;
```

```
2) SELECT eh.patient_id, es.value
FROM event_header eh LEFT JOIN
     (SELECT es.header_uid, er.value from EVENT_SUBHEADER es, eav_real er
      WHERE es.subheader_id = er.subheader_id AND er.question_id = 161) es
    ON eh.header_uid = es.header_uid
WHERE  eh.study_id = 3 AND eh.cluster_id = 65;
```

Cvičení

•Vytvořte z vnořeného dotazu VIEW a přepište předchozí dotaz s jeho použitím

```
CREATE VIEW subheader_eav
as
SELECT es.header_uid, er.value FROM event_subheader es, eav_real er
      WHERE es.subheader_id = er.subheader_id AND er.question_id = 161
```

```
SELECT eh.patient_id, es.value
FROM event_header eh LEFT JOIN subheader_eav es
    ON eh.header_uid = es.header_uid
WHERE  eh.study_id = 3 AND eh.cluster_id = 65
```

Cvičení

Lepší varianta umožňující využití pro libovolnou otázku

```
CREATE OR REPLACE VIEW subheader_eav
as
SELECT es.header_uid, er.question_id, er.value
FROM event_subheader es, eav_real er
      WHERE es.subheader_id = er.subheader_id
```

```
SELECT eh.patient_id, es.value
FROM event_header eh LEFT JOIN subheader_eav es
    ON eh.header_uid = es.header_uid  AND es.question_id = 161
WHERE  eh.study_id = 3 AND eh.cluster_id = 65
```

Cvičení

- Vypište hodnoty dvou otázek (161, 27) ve tvaru patient_id, value1, value2 pro všechny existující formuláře cluster_id = 65 , study_id = 3

```
SELECT eh.patient_id, es.value, es2.value
FROM event_header eh
    LEFT JOIN subheader_eav es
      ON eh.header_uid = es.header_uid and es.question_id = 161
    LEFT JOIN subheader_eav es2
      ON eh.header_uid = es2.header_uid and  es2.question_id = 27
WHERE  eh.study_id = 3 AND eh.cluster_id = 65
```

ORACLE varianta:

```
SELECT eh.patient_id, es.value, es2.value
FROM event_header eh, subheader_eav es, subheader_eav es2
     WHERE eh.header_uid = es.header_uid(+) and es.question_id(+) = 161
     AND eh.header_uid = es2.header_uid(+) and  es2.question_id(+) = 27
AND  eh.study_id = 3 AND eh.cluster_id = 65
```

Cvičení

- Vypište formuláře (cluster_id = 65 , study_id = 3), které nemají vyplněny otázky question_id 161, 27 ve tvaru patient_id, header_uid

```
SELECT eh.patient_id, eh.header_uid
FROM   event_header eh
WHERE  eh.study_id = 3 AND eh.cluster_id = 65
AND  NOT EXISTS
   (SELECT * FROM subheader_eav es
      WHERE eh.header_uid = es.header_uid AND es.question_id = 161
   )
AND NOT  EXISTS
   (SELECT * FROM  subheader_eav es2
      WHERE  eh.header_uid = es2.header_uid AND  es2.question_id = 27
   )
```

Domácí úkol

- V PosgreSQL databázi importujte další skript