

# Normalization Requirements

**Relational Databases Basics** 



Before we begin...

It is extremely useful to remember that during normalization process we can achieve both important goals:

- eliminate data operation anomalies;
- significantly improve the whole database quality.

We shall discuss it now.

#### Disclaimer

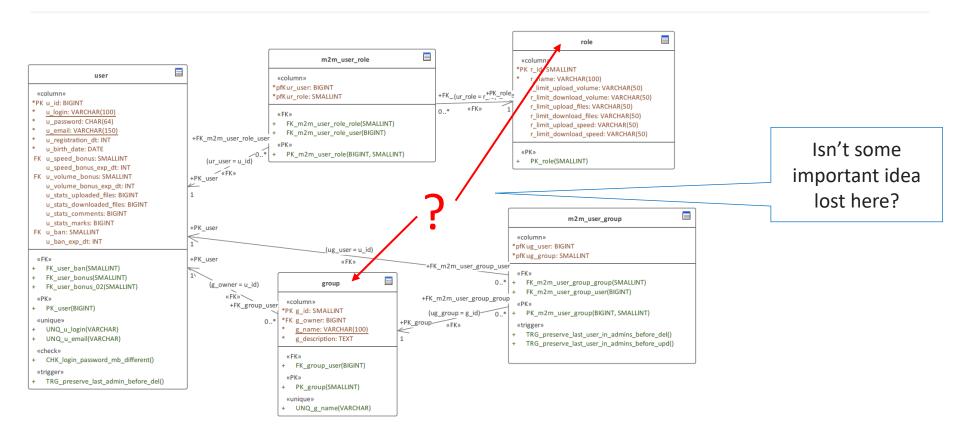
In the next several minutes we shall review a lot of so called "normalization requirements". It's hard to comply with them all in a single database, but this is not the goal.

The idea is to determine our main quality goals and follow those "normalization requirements" that bring us closer to that goals.

**Normalization** – a process of decomposition of relation variable R into projections  $R_1$ ,  $R_2$ , ...,  $R_n$ , such that:

- the join of R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub> is guaranteed to be equal to R;
- each of R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub> is needed in order to provide that guarantee;
- at least one of R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub> is at a higher level of normalization than R is.

# Subject matter constraints preservation





# Fundamental database requirements compliance

Subject matter adequacy

3 Performance

Technical usability

Data safety

# Primary keys minimality

Does a table need a Primary Key at all?

Is this Primary Key small enough?

Is this Primary Key big enough?

Can we preserve data range making the Primary Key smaller?

# Data storage non-redundancy

#### contract

•••	c_serial_number	c_sum	•••
	AC345347856DF	34 000 000	•••
	DF345345652YH	12 000 000	
	AA345235235KL	32 511 012	
	GT456345342UT	41 356 343	

#### finance

•••	f_serial_number	f_sum	•••
•••	AC345347856DF	29 627 532	
•••	DF345345652YH	12 000 000	
•••	AA345235235KL	32 511 012	
•••	GT456345342UT	41 356 343	

# General database performance

0.1-0.2 reads, 200-300K reads, 500-700K writes. news log 5-10 writes, 0 updates «column» «column» 2-3 updates \*PK n\_id: INT I datetime: INT n\_rubric: VARCHAR(100) I operation: VARCHAR(50) n datetime: INT I paremeters: VARCHAR(250) n title: VARCHAR(250) n annotation: VARCHAR(10000) n author: VARCHAR(100) 1-2 reads, aggregation n text: TEXT 1 write, n\_source: VARCHAR(250) «column» 0 updates \*PK a date: DATE «PK» \*PK a operation type: VARCHAR(50) PK\_news(INT) a\_operation\_count: INT «index» IDX\_rubric\_dt(VARCHAR, INT) «PK» IDX title dt(VARCHAR, INT) PK\_aggregation(DATE, VARCHAR) IDX\_dt(INT) IDX\_author\_rubric(VARCHAR, VARCHAR)

Data consistency and integrity

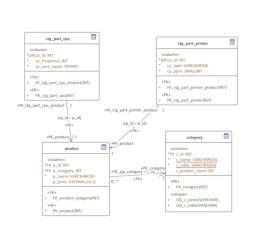
Create explicit relationships

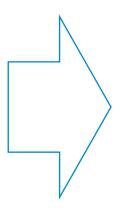
Use triggers, checks, any other "constraint mechanisms"

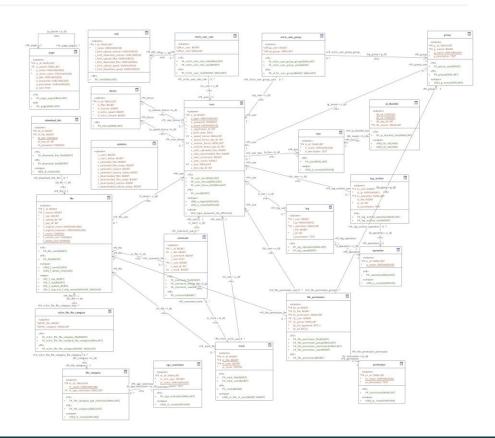
Mind cache invalidation

All consistency and integrity constraints should be explicitly defined on database (NOT application) side!

# Database flexibility through ...







# Database flexibility through naming convention

	BAD	GOOD	
	data	registered_user	
Table names	connection	m2m_user_role	
	t1	news_rubric	
	рр	u_primary_phone	
Filed names	ok	o_is_order_confirmed	
	some_text_data	a_biography	
	process	get_initials	
Stored subroutine names	get_date	unixtime_to_datetime	
	fnc	upcase_first_letters	
	no_admins	t_user_protect_last_admin_del	
Trigger names	update_all	t_aggregate_user_stats_upd	
	bad_dates	t_date_start_le_end_ins_upd	
	qsrch	idx_rubric_author_date	
Index names	singlerecord	unq_login	
	tr	idx_author_btree	

### Database flexibility through comments

```
CREATE TABLE `file`
 `f uid`
                     BIGINT UNSIGNED NOT NULL AUTO INCREMENT
                     COMMENT 'Global file identifier.',
 `f fc uid`
                     BIGINT UNSIGNED NULL
                     COMMENT 'File category identifier (FK to "category" table).',
 `f size`
                     BIGINT UNSIGNED NOT NULL
                     COMMENT 'File size (bytes).',
 `f upload datetime` INTEGER NOT NULL
                     COMMENT 'File upload datetime (Unixtime).',
 `f save datetime`
                    INTEGER NOT NULL
                     COMMENT 'File expiration datetime (Unixtime).',
 `f src name`
                     VARCHAR (255) NOT NULL
                     COMMENT 'Initial file name (as it was on user device). With no
                              extension, as it is stored separately in "f src ext" field!',
 `f src ext`
                     VARCHAR (255) NULL
                     COMMENT 'Initial file extension (as it was on user device).',
 `f name`
                     CHAR (200) NOT NULL
                     COMMENT 'Stored server-side file name. Five SHA1-hashes.',
 `f sha1 checksum`
                     CHAR (40) NOT NULL
                     COMMENT 'File control sum, SHA1-hash.',
 `f ar uid`
                     BIGINT UNSIGNED NOT NULL
                     COMMENT 'File access permissions (FK to "access rights" table).',
 `f downloaded`
                     BIGINT UNSIGNED NULL DEFAULT 0
                     COMMENT 'File downloads counter.',
```

# Database flexibility through documentation

Schemas and comments are good, but some text explanations are good too.

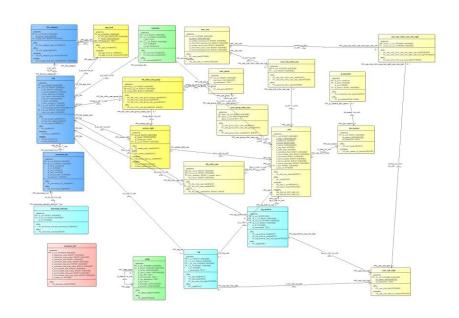


# Database flexibility through schemas

# Auto-generated

#### user\_role\_m2m\_user\_role\_right \* fmug\_ug\_uid B3S3NT(20) I u uid BISINT (20) 1 ib. ipv6 CHAR(39) ofc name VARCHAR(255) I\_urr\_uid INT(30) ofmug\_rights 8303NT(20) Ofc\_age\_limit B3GDVT(20) ◆ c\_f\_uid 83GDHT(20) c\_text TEXT ◆ c. u. uid BIGINT(20 oc\_dt INT(11) ur\_uid 3NT(s0) \* ar\_uid 81G (NT (20) ar\_name VARCHAR (255) of\_ft\_uid BOSENT(20) u\_login WARCHAR(180) ri\_registered\_users\_today 83GBVT(20 or\_description TEXT == u\_erral 53'4RY(255) ri\_uploaded\_files\_today 83G3NT(20) ur\_upload\_count\_limit 80GINT(20) f\_save\_datetine INT(11) u\_name VARCHAR(255) ri uploaded files size BIGINTIZED ur download count limit 86G3NTT201 f grc name VAROHAR(255 u reg date INT(11) r uploaded files size today BIGINT(20) of ac ext VAROHAR(255) u birth date INT(11) i doorloaded files EDSINT (20) f name (24A0(200) u\_uploaded\_files B3GZNT(20) r\_downloaded\_files\_today 80GINT (20) f\_shal\_checksum Q1AR(4 u\_downloaded\_files 81GINT(2) r\_downloaded\_files\_size\_80G3NT(20) ◆ f\_ar\_uid BIGINT(20) u\_comments B3GINT(20) of\_downloaded BEEENT(20) u\_bonus\_speed BIGENT(20) I will id SIGINT(20) Of all aid RECONT (20) Our bonus unload RIGINT(20) al\_min\_age SMALLINT(S) f del link hash CHARC u bonus download BIGINT(2) al\_name VARCHAR(255) ou\_ban BOGENT(20) u\_ban\_dt DVT(LL) od hash CHAR(240) @df\_dt INT(11) Odl lov4 CHAR(15) Od\_lov6 CHAR(39) od\_cooke\_hash O14R(240 od\_password OHAR(40) 0-61\_dR INT(11) On parent D/T (10) p name VARCHAR (255) p\_menu\_name VARCHARIZSS p\_hdr\_title TEXT p\_hdr\_keywords TEXT p\_hdr\_description TEXT Fugmu up uid 80 GENT (20 fmu\_f\_uid 81G1NT(20) la.uld BEWARY(240) fugmu\_u\_uid 86G3NT(20) ug\_name VARCHAR(255) Obr\_name VARCHAR(180) p\_test TEXT fmu\_u\_uid 81GENT(20 Ola\_u\_uid 80G3NT(20) uonu is owner BIT(1) p template VARCHAR(255) frou direction ENUMS... ♦ la\_urr\_uid DVF(10) o p. handler VARCHAR(255) In f. uid BOGDNT (20) la\_dt INT(11)

# Manually created



# Database flexibility through logical approach

Imagine you see such a code in database creation script...

```
ALTER TABLE `file`

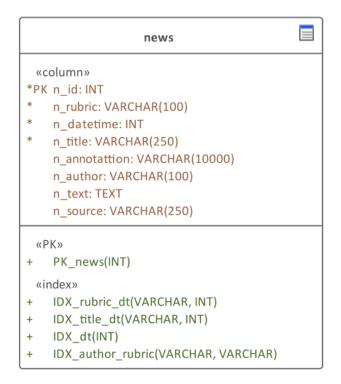
ADD INDEX `IDX_f_orig_ext_f_orig_name`
(`f_original_extension`(200) ASC,`f_original_name`(200) ASC);

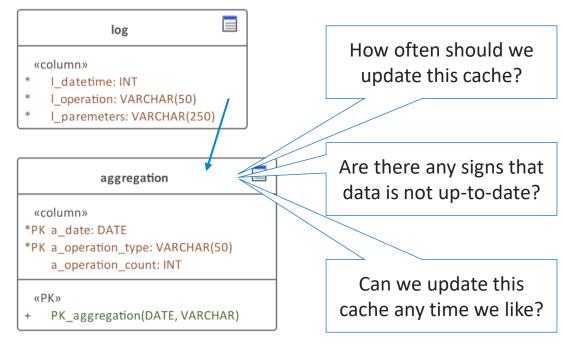
????

????
```

Are these limitations necessary? Isn't it s mistake? Where are comments?!?! Shall we leave this intact or change it? Will this change break anything?

### Data relevance (up-to-date state)







# Normalization Requirements

**Relational Databases Basics** 

